cadence®

Innovus Stylus Common UI Text Command Reference

Product Version 22.13
Last Updated in July 2023

© 2023 Cadence Design Systems, Inc.
Printed in the United States of America.

Cadence Design Systems, Inc. (Cadence), 2655 Seely Ave., San Jose, CA 95134, USA.

Open SystemC, Open SystemC Initiative, OSCI, SystemC, and SystemC Initiative are trademarks or registered trademarks of Open SystemC Initiative, Inc. in the United States and other countries and are used with permission.

Trademarks: Trademarks and service marks of Cadence Design Systems, Inc. (Cadence) contained in this document are attributed to Cadence with the appropriate symbol. For queries regarding Cadence's trademarks, contact the corporate legal department at the address shown above or call 800.862.4522.

All other trademarks are the property of their respective holders.

Restricted Permission: This publication is protected by copyright law and international treaties and contains trade secrets and proprietary information owned by Cadence. Unauthorized reproduction or distribution of this publication, or any portion of it, may result in civil and criminal penalties. Except as specified in this permission statement, this publication may not be copied, reproduced, modified, published, uploaded, posted, transmitted, or distributed in any way, without prior written permission from Cadence. Unless otherwise agreed to by Cadence in writing, this statement grants Cadence customers permission to print one (1) hard copy of this publication subject to the following conditions:

1. The publication may be used only in accordance with a written agreement between Cadence and its customer.
2. The publication may not be modified in any way.
3. Any authorized copy of the publication or portion thereof must include all original copyright, trademark, and other proprietary notices and this permission statement.
4. The information contained in this document cannot be used in the development of like products or software, whether for internal or external use, and shall not be used for the benefit of any other party, whether or not for consideration.

Disclaimer: Information in this publication is subject to change without notice and does not represent a commitment on the part of Cadence. Except as may be explicitly set forth in such agreement, Cadence does not make, and expressly disclaims, any representations or warranties as to the completeness, accuracy or usefulness of the information contained in this document. Cadence does not warrant that use of such information will not infringe any third party rights, nor does Cadence assume any liability for damages or costs of any kind that may result from use of such information. Cadence is committed to using respectful language in our code and communications. We are also active in the removal and replacement of inappropriate language from existing content. This product documentation may however contain material that is no longer considered appropriate but still reflects long-standing industry terminology. Such content will be addressed at a time when the related software can be updated without end-user impact.

Restricted Rights: Use, duplication, or disclosure by the Government is subject to restrictions as set forth in FAR52.227-14 and DFAR252.227-7013 et seq. or its successor.

Contents

About This Manual 106
Audience 106
Conventions Used in This Manual 106
Related Documents 108
Innovus Product Documentation 108
Innovus Stylus Common UI Documentation 109
1 110
Advanced Timing Tcl Scripting Commands 110
Overview 111
all_* Commands 111
Examples 112
get_* Commands 112
Examples 112
property Commands 113
Examples 113
collection Commands 113
Examples 114
add_to_collection 115
all_clocks 116
all_connected 117
all_fanin 118
all_fanout 121
all_inputs 124
all_instances 125
all_outputs 126
all_registers 127
append_to_collection 129
compare_collections 131
copy_collection 132
define_property 133
filter_collection 135
foreach_in_collection 137

get_activity 139
get_arcs 144
get_cells 146
get_clocks 149
get_designs 151
get_generated_clocks 152
get_lib_arcs 153
get_lib_cells 155
get_lib_pg_pins 157
get_lib_pins 159
get_libs 161
get_nets 163
get_object_name 165
get_path_groups 166
get_pg_nets 167
get_pg_pins 169
get_pins 170
get_ports 172
get_power 174
get_property 178
261
index_collection 264
query_objects 265
range_collection 266
remove_from_collection 268
report_object_property_names 269
report_property 270
sizeof_collection 272
sort_collection 273
2 275
ART Based Flow Commands 275
clear_active_logic_view 275
create_active_logic_view 276
get_module_view 277
set_module_view 279
3 281

Basic Database Access Tcl Commands and Attributes 281
add_line_end_track 281
commit_global_net_rules 283
compress_special 284
connect_global_net 285
convert_dbu_to_um 288
convert_um_to_dbu 289
create_cell_obs 290
create_via 291
create_via_definition 292
create_virtual_shape_on_pg_wires 297
decompress_special 298
delete_cell_obs 299
delete_floating_nets 300
delete_global_net_connections 301
get_computed_shapes 302
get_edge 305
get_obj_in_area 306
get_rect 309
get_transform_shapes 311
report_floating_nets 313
report_metal_stack 315
report_net_wires 316
Node Information 316
Edge Information 316
report_preserves 318
report_rc 320
report_via_def 324
report_wire_path 326
safety Category Attributes 327
safety_flow_enable 327
safety_midas_enable 327
safety_ser_cells 328
safety_tmr_isolate_clock 328
safety_tmr_spacing 328
safety_tmr_spacing_x 328

safety_tmr_spacing_y 328
tech Category Attributes 328
tech_db_units 329
tech_finfet_grid_direction 329
tech_finfet_grid_offset 329
tech_finfet_grid_pitch 329
tech_inst_mask_shift_layers 329
tech_mfg_grid 329
uniquify 330
4 333
Bus Plan Commands 333
create_bus_guide 333
delete_bus_guides 337
gui_clear_bus_guide_colors 339
gui_show_bus_guide_multi_colors 340
update_bus_guide 342
5 346
Check Commands and Attributes 346
check_ac_limit Category Attributes 346
check_ac_limit_additional_individual_violation_report 348
check_ac_limit_avg_recovery 348
check_ac_limit_current_file 348
check_ac_limit_current_scale_factor 348
check_ac_limit_current_scale_table 349
check_ac_limit_default_freq_for_unconstrained_nets 349
check_ac_limit_delta_temperature 349
check_ac_limit_delta_temperature_layer_list 349
check_ac_limit_detailed 349
check_ac_limit_effort_level 349
check_ac_limit_em_cdf_percentage 350
check_ac_limit_em_limit_scale_factor 350
check_ac_limit_em_limit_scale_table 350
check_ac_limit_em_res_width 350
check_ac_limit_em_temperature 351
check_ac_limit_em_temperature_layer_list 351
check_ac_limit_em_threshold 351

check_ac_limit_enable_seb 351
check_ac_limit_env_temperature 351
check_ac_limit_extraction_tech_file 351
check_ac_limit_force_hold_view 351
check_ac_limit_handle_pin_obs_via 352
check_ac_limit_ict_em_models 352
check_ac_limit_ircx_models 352
check_ac_limit_lifetime 352
check_ac_limit_max_error 352
check_ac_limit_method 353
check_ac_limit_min_peak_duty_ratio 353
check_ac_limit_min_peak_freq 353
check_ac_limit_net_file 353
check_ac_limit_nets 353
check_ac_limit_out_file 353
check_ac_limit_peak_td_method 354
check_ac_limit_seb_lifetime 355
check_ac_limit_seb_table 355
check_ac_limit_seb_temperature 356
check_ac_limit_selected 356
check_ac_limit_simulation_net 356
check_ac_limit_simulation_net_file 356
check_ac_limit_skip_category_mode 356
check_ac_limit_skip_net 357
check_ac_limit_skip_net_file 357
check_ac_limit_toggle 357
check_ac_limit_use_db_freq 358
check_ac_limit_use_qrc_tech 358
check_ac_limit_use_rms_delta_t 358
check_ac_limit_view 358
check_ac_limits 358
check_antenna 360
check_connectivity 362
check_cut_density 366
check_drc 368
check_drc Category Attributes 371
check_drc_area 371

check_drc_check_only 371
check_drc_check_reverse 372
check_drc_check_routing_halo 372
check_drc_check_routing_halo_corner 372
check_drc_disable_rules 373
check_drc_enable_post_passive_fill 373
check_drc_exclude_pg_net 373
check_drc_ignore_cell_blockage 373
check_drc_ignore_non_rectangle_shapes 373
check_drc_ignore_trial_route 374
check_drc_inside_via_def 374
check_drc_layer_range 374
check_drc_limit 374
check_drc_max_wrong_way_halo 375
check_drc_ndr_spacing 375
check_drc_report 375
check_drc_short_only 375
check_drc_trim_length 376
check_drc_uncolored 376
check_drc_use_min_spacing_on_block_obs 376
check_endcaps 376
check_io_to_bump_connectivity 378
check_isolated_cuts 379
check_litho 381
check_metal_density 384
check_pg_shorts 387
check_power_vias 388
check_process_antenna 394
check_tieoffs 396
check_well_antenna 397
check_well_taps 398
check_wire_gaps 401
create_marker 403
delete_drc_markers 405
delete_markers 406
get_signoff_verify_design_config 407
gui_show_markers 409

opt_pattern 412
read_drc_markers 416
read_markers 417
report_markers 420
set_isolated_cut_rule 422
set_signoff_verify_design_config 423
set_snap_grid 428
signoff_verify_design 429
write_drc_markers 434
6 435
Clock Tree Synthesis Commands and Attributes 435
add_clock_tree_exclusion_drivers 436
add_clock_tree_source_group_roots 438
assign_clock_tree_source_groups 440
ccopt_design 441
ccopt Category Attributes 442
ccopt_auto_limit_insertion_delay_factor 442
ccopt_merge_clock_gates 442
ccopt_merge_clock_logic 442
check_clock_tree_convergence 443
check_cts_config 444
clock_design 445
clock_opt_design 446
clock_post_route_repair 448
commit_clock_tree_route_attributes 450
connect_clock_tree_mesh_drivers 451
convert_lib_clock_tree_latencies 452
create_clock_tree 456
create_clock_tree_source_group 457
create_clock_tree_spec 458
create_flexible_htree 460
create_generated_clock_tree 467
create_route_type 469
create_skew_group 471
cts Category Attributes 474
clock_trees 476

clock_tree_source_groups 476
cts_add_port_driver_max_distance_from_port 477
cts_add_port_driver_max_distance_from_port_rows 477
cts_adjacent_rows_legal 477
cts_allow_non_std_clock_gate_inputs 477
cts_balance_mode 478
cts_blackbox_default_driver_base_pin 478
cts_blackbox_default_load_base_pin 478
cts_buffer_cells 478
cts_buffer_cells_leaf 478
cts_buffer_cells_top 479
cts_cell_density 479
cts_cell_halo_mode 480
cts_cell_halo_rows 480
cts_cell_halo_sites 480
cts_cell_halo_x 481
cts_cell_halo_y 481
cts_clock_gating_cells 482
cts_clock_gate_movement_limit 482
cts_clock_source_cells 482
cts_clock_tree_source_group_clock_trees 482
cts_clock_tree_source_input_max_transition_time 483
cts_cloning_inst_name_suffix 483
cts_cloning_inst_name_suffix_source_group_assignment 483
cts_delay_cells 483
cts_detailed_cell_warnings 483
cts_exit_if_no_placeable_area 484
cts_exit_if_skew_target_over_constrained 484
cts_exit_if_stage_delay_sigma_target_over_constrained 484
cts_fix_clock_sinks 484
cts_flexible_htree_enable_metrics_reporting 484
cts_flexible_htree_partition_boundary_polarity 484
cts_flexible_htree_placement_legalization_effort 485
cts_force_target_max_transition_time 485
cts_ignore_problematic_skew_as_result_of_dont_touch_nets 485
cts_inst_name_prefix 485
cts_inverter_cells 485

cts_inverter_cells_leaf 486
cts_inverter_cells_top 486
cts_load_capacitance_cells 486
cts_logic_cells 486
cts_manage_power_intent_violations 487
cts_max_fanout 487
cts_max_instance_height 487
cts_max_source_to_sink_net_length 487
cts_max_source_to_sink_net_length_leaf 487
cts_max_source_to_sink_net_length_top 488
cts_max_source_to_sink_net_length_trunk 488
cts_merge_clock_gates 488
cts_merge_clock_logic 488
cts_mixed_fanout_net_type 489
cts_move_clock_gates 489
cts_move_logic 489
cts_net_name_prefix 489
cts_override_vias 489
cts_pin_insertion_delay_histogram_bin_size 490
cts_post_route_enable 490
cts_post_route_repair_drv 490
cts_post_route_enable_routing_eco 490
cts_post_route_repair_skew_by_buffering 490
cts_primary_delay_corner 490
cts_primary_reporting_skew_groups 491
cts_primary_reporting_skew_groups_log_min_max_sinks 491
cts_repair_congestion 491
cts_repair_drv_by_buffering 491
cts_report_skew_groups_only_with_targets 491
cts_route_clock_tree_nets 492
cts_route_type_leaf 492
cts_route_type_top 492
cts_route_type_trunk 492
cts_routing_preferred_layer_effort 492
cts_size_clock_gates 493
cts_size_clock_sources 493
cts_size_logic 493

cts_skew_group_report_columns 493
cts_skew_group_report_histogram_bin_size 494
cts_spec_config_base_pin_trace_through_to 494
cts_spec_config_create_clock_tree_source_groups 494
cts_spec_config_create_generator_skew_groups 495
cts_spec_config_create_generator_skew_groups_name_prefix 495
cts_spec_config_create_reporting_only_skew_groups 495
cts_spec_config_trace_multi_output_cells_with_single_clock_output 495
cts_target_max_capacitance 496
cts_target_max_capacitance_leaf 496
cts_target_max_capacitance_top 496
cts_target_max_capacitance_trunk 497
cts_target_max_stage_delay_sigma 497
cts_target_max_transition_time 497
cts_target_max_transition_time_leaf 497
cts_target_max_transition_time_top 497
cts_target_max_transition_time_trunk 498
cts_target_skew 498
cts_timing_connectivity_based_skew_groups_balance_master_clocks 498
cts_timing_connectivity_based_skew_groups 498
cts_timing_connectivity_info 499
cts_top_fanout_threshold 499
cts_update_clock_latency 499
cts_use_inverters 500
cts_use_receiver_model_capacitance_for_drv 500
flexible_htrees 500
skew_groups 500
delete_clock_tree_drivers 501
delete_clock_tree_source_group 502
delete_clock_tree_spec 503
delete_clock_trees 504
delete_flexible_htrees 505
delete_skew_groups 506
get_ccopt_property_common_ui_map 507
get_skew_group_path 510
get_clock_tree_capacitance 513
get_clock_tree_effective_max_capacitance 515

get_clock_tree_transition_time 517
get_cts_graph_traversal 518
get_lib_clock_tree_path_delay 520
get_skew_group_delay 522
gui_ctd_close 526
gui_ctd_get_title 527
gui_ctd_get_window_id 528
gui_ctd_highlight 529
gui_ctd_open 532
gui_ctd_write_histogram 535
gui_ctd_write_view 536
gui_ctd_set_title 537
gui_ctd_zoom 539
merge_clock_tree_instances 542
place_detail_clock_tree 545
read_cts_config 547
report_cts_cell_name_info 548
report_ccopt_worst_chain 549
report_clock_tree_convergence 551
report_clock_tree_halo_violations 552
report_clock_tree_structure 555
report_clock_trees 559
report_cts_cell_filtering_reasons 563
report_pin_insertion_delays 566
report_skew_groups 569
reset_cts_config 573
reset_clock_tree_routing_state 574
route_flexible_htrees 575
route_clock_tree_nets 576
synthesize_flexible_htrees 577
update_clock_tree_source_latency 580
update_clock_tree_spec_annotations 581
update_skew_group 583
write_cts_config 584

7 586

Conformal Commands 586

run_clp 586
8 589
Database Access and Handling Commands and Attributes 589
define_attribute 589
delete_buffer 593
delete_obj 594
get_db 595
get_db Category Attributes 604
get_db_display_limit 604
is_attribute 604
reset_db 606
set_db 609
sort_db 611
write_preserves 613
9 615
Delay Calculation Commands and Attributes 615
add_decap_cell_candidates 615
add_decaps 617
delaycal Category Attributes 621
delaycal_accuracy_level 622
delaycal_advanced_pin_cap_mode 623
delaycal_advanced_node_pin_cap_settings 623
delaycal_combine_mmmc 623
delaycal_degrade_slew_on_early_nets 624
delaycal_default_net_delay 624
delaycal_default_net_load 624
delaycal_enable_high_fanout 624
delaycal_enable_quiet_receivers_for_hold 625
delaycal_enable_si 625
delaycal_enable_wire_load_model 625
delaycal_equivalent_waveform_model 626
delaycal_equivalent_waveform_type 626
delaycal_equivalent_waveform_model_for_timing_check 626
delaycal_honor_slew_propagate_constraint 626
delaycal_ignore_net_load 627
delaycal_input_transition_delay 627

delaycal_irdrop_data_type 628
delaycal_early_irdrop_data_type 628
delaycal_late_irdrop_data_type 628
delaycal_library_interpolation_mode 628
delaycal_irdrop_window_based 629
delaycal_report_out_bound 629
delaycal_signoff_alignment_settings 629
delaycal_slew_out_bound_limit_high 629
delaycal_slew_out_bound_limit_low 630
delaycal_skip_slew_merge_from_disabled_path 630
delaycal_socv_accuracy_mode 630
delaycal_socv_lvf_mode 630
delaycal_socv_machine_learning_level 631
delaycal_socv_use_lvf_tables 631
delaycal_support_output_pin_cap 631
delaycal_support_wire_load_model 631
delaycal_timing_create_clock_use_ideal_slew 632
delaycal_use_default_delay_limit 632
delete_decaps 632
delete_exclude_net_attribute 634
report_decap_cell_candidates 635
report_decaps 637
report_delay_calculation 638
reset_decap_cell_candidates 645
write_exclude_nets 646
write_set_load 647
10 649
Flip Chip Commands and Attributes 649
assign_bumps 650
assign_pg_bumps 654
assign_signal_to_bump 659
check_bond_pad_spacing 660
check_bumps 661
check_flip_chip_constraints 674
check_io_inst_overlap 675
copy_bumps 676

create_bump 678
create_bump_connect_target_constraint 694
create_signal_pin 696
delete_area_io_fillers 698
delete_bump_connect_target_constraints 699
delete_bumps 701
flip_chip Category Attributes 703
- flip_chip_allow_layer_change 704
- flip_chip_allow_routed_bump_edit 704
- flip_chip_auto_pairing_file 704
- flip_chip_bottom_layer 705
- flip_chip_bump_use_octagon_shape 705
- flip_chip_check_bump_access_directions 705
- flip_chip_compaction 706
- flip_chip_connect_power_cell_to_bump 706
- flip_chip_constraint_file 707
- flip_chip_drop_via_on_all_shapes 707
- flip_chip_drop_via_on_power_mesh 708
- flip_chip_extra_config 709
- flip_chip_finger_direction 709
- flip_chip_finger_max_width 709
- flip_chip_finger_min_width 709
- flip_chip_finger_target_mesh_layer_range 710
- flip_chip_honor_bump_connect_target_constraint 710
- flip_chip_ignore_pad_type_check 710
- flip_chip_lower_layer_route_width 710
- flip_chip_lower_layer_prevent_diagonal_routing 711
- flip_chip_multi_pad_routing_style 711
- flip_chip_multiple_connection 712
- flip_chip_pg_mesh_direction 712
- flip_chip_pg_mesh_main_width 713
- flip_chip_pg_mesh_max_width 713
- flip_chip_pg_mesh_width 713
- flip_chip_prevent_diagonal_wire_access_bumps 714
- flip_chip_prevent_via_under_bump 714
- flip_chip_prevent_via_under_bump_extension 714
- flip_chip_route_pg_style 716

flip_chip_route_style 716
flip_chip_route_width 716
flip_chip_top_layer 716
flip_chip_via_abut_bump 717
flip_chip_via_to_pad_honor_min_spacing 717
flip_chip_wire_to_pad_honor_min_spacing 717
get_bond_pad 718
get_physical_info 719
get_pin_port_number 721
gui_show_bump_connections 724
place_bond_pad 727
place_bond_pad_by_stagger 728
read_bump_color_map_file 732
read_flip_chip_property 733
read_io_update 734
read_package 735
report_probe_pins 736
report_special_routes 737
reset_bumps_fixed_assign 743
reset_fixed_bond_pad 744
reset_probe_pin 745
route_flip_chip 746
route_point_to_point 753
set_bond_pads_to_fixed 756
set_bump_master 757
set_bumps_fixed_assign 759
set_bumps_place_status 760
set_probe_pin 761
set_special_route_type 762
space_bond_pads 764
swap_bump_assignment 766
unassign_bumps 767
update_bump_connect_target_constraints 768
update_bump_name 770
write_flip_chip_property 771
11 772

Floorplan Commands and Attributes 772
add_area_io_fillers 775
add_area_io_rows 777
add_dummy_boundary_wires 780
add_fences 783
add_interposer_route_block 786
add_io_fillers 787
add_io_insts 789
add_io_row_fillers 791
add_physical_cell 794
align_selected 798
check_floorplan 801
check_floorplan_space 802
check_macro_lower_left_on_track 803
check_macro_place_constraint 804
check_ufc 805
convert_fence_to_lef 808
copy_obj 809
create_boundary_constraint 810
create_bump_grid 812
create_exclusive_groups 813
create_floorplan 815
create_group 825
create_interposer_route_block 827
create_io_row 828
create_logic_hierarchy 831
create_pg_model_for_macro_place 833
create_pg_pin 834
create_physical_pin 837
create_place_blockage 839
create_place_halo 844
create_relative_floorplan 846
create_resize_blockage 851
create_route_blockage 852
create_route_blockage_along_track 856
create_route_halo 859

create_route_rule 862
create_row 865
create_shape 867
create_special_drc_region 869
create_special_mark_layer_shape 871
create_text 873
delete_all_density_area 875
delete_all_floorplan_objs 876
delete_all_power_preroutes 877
delete_all_signal_preroutes 878
delete_bump_grid 879
delete_exclusive_groups 880
delete_io_fillers 881
delete_io_insts 882
delete_io_row_fillers 883
delete_pg_pins 884
delete_place_halo 885
delete_relative_floorplan 886
delete_route_blockages 887
delete_route_halos 888
delete_row 889
delete_selected_from_floorplan 890
finish_floorplan 891
finish_floorplan Category Attributes 902
- finish_floorplan_abutted_edge_as_non_row_area 902
- finish_floorplan_active_objs 903
- finish_floorplan_add_blockage_direction 903
- finish_floorplan_drc_region_objs 903
- finish_floorplan_override 904

fix_boundary_overlaps 905
fix_floorplan 906
flip_or_rotate_obj 908
floorplan Category Attributes 910
- floorplan_auto_sync_master_clone 911
- floorplan_cells_for_extra_sites 911
- floorplan_check_types 911
- floorplan_cut_off_place_blockage_outside_die 912

floorplan_cut_off_route_blockage_outside_die 912
floorplan_default_blockage_name_prefix 913
floorplan_default_power_domain_site 913
floorplan_default_tech_site 913
floorplan_enable_rectilinear_design 913
floorplan_extra_sites 914
floorplan_force_honor_snap_rule 914
floorplan_include_io_when_init_area 914
floorplan_initial_all_compatible_core_site_rows 914
floorplan_keep_rows_when_moving_power_domain 915
floorplan_max_io_height 915
floorplan_minimum_sites 915
floorplan_move_macros_with_constraint 915
floorplan_move_preplaced_std_cell_only 915
floorplan_move_std_cell_with_constraint 916
floorplan_narrow_channel_threshold 916
floorplan_no_cut_row 916
floorplan_finfet_inst_grid 916
floorplan_finfet_manufacturing_grid 917
floorplan_finfet_placement_grid 917
floorplan_inst_grid 917
floorplan_layer_track_grid 918
floorplan_manufacturing_grid 918
floorplan_placement_grid 918
floorplan_power_rail_layer 919
floorplan_row_site_height 919
floorplan_row_site_width 919
floorplan_snap_all_corners_to_grid 919
floorplan_snap_block_grid 920
floorplan_snap_constraint_grid 920
floorplan_snap_core_grid 921
floorplan_snap_die_grid 921
floorplan_snap_io_grid 921
floorplan_snap_place_blockage_grid 922
floorplan_snap_place_blockage_type 922
floorplan_user_define_grid 923
floorplan_vertical_row 923

get_draw_view 924
get_io_flow_flag 925
get_macro_place_constraint 926
gui_set_draw_view 929
gui_show_hinst_in_floorplan 930
gui_show_obj_connectivity 931
init_core_rows 933
legalize_floorplan 934
move_group_pins 935
move_macro_in_hinsts 937
move_obj 938
paste_obj 940
place_area_ios 941
place_blocks_refine 943
place_macro_detail 945
place_pad_ios 946
place_peripheral_ios 947
plan_design Category Attributes 950
  plan_design_domain 950
  plan_design_rail_model 950
  plan_design_total_power 950
read_floorplan 951
read_io_file 953
report_floorplan_obj 954
report_narrow_channel 955
report_net_groups 958
report_obj_connectivity 959
report_off_grid_blocks 961
report_seed_connections 962
reset_fixed_module_size 963
reset_macro_place_constraint 964
resize_floorplan 966
resize_floorplan_channel 968
resize_floorplan Category Attributes 970
  resize_floorplan_cong_aware 970
  resize_floorplan_honor_halo 970
  resize_floorplan_io_fix 970

resize_floorplan_io_move_with_edge 971
resize_floorplan_io_relative 971
resize_floorplan_maintain_resource_ratio 971
resize_floorplan_relative 971
resize_floorplan_shift_based 972
resize_floorplan_shrink_fence 972
resize_floorplan_snap_to_track 972
resize_partition_to_abut 973
set_all_ios_to_fixed 979
set_bottom_io_pad_orient 980
set_flipping 981
set_floorplan_row_spacing_and_type 982
set_inst_group_physical_hier 983
set_instance_placement_status 984
set_io_constraints 985
set_io_flow_flag 988
set_io_row_margin 989
set_macro_place_constraint 990
set_module_size_fixed 998
set_resize_line 999
set_route_blockage_default_layer 1001
snap_floorplan 1003
snap_floorplan_io 1005
space_selected 1006
split_rectilinear_inst 1008
split_row 1011
stretch_rows 1013
swap_pins 1014
trace_obj_connectivity 1015
trace_obj Category Attributes 1017
trace_obj_macro_pins 1017
trace_obj_max_fanin_fanout 1017
trace_obj_register_inputs 1017
trace_obj_register_outputs 1017
unplace_obj 1019
update_floorplan 1020
update_floorplan_for_macro_place 1021

update_floorplan_obj 1022
update_group 1023
update_obj_floorplan_cut_box 1024
update_origin 1025
update_partition_to_stairway 1029
update_route_rule 1032
write_floorplan 1035
write_floorplan_script 1037
write_io_file 1040
write_macro_place_constraint 1043
write_route_rule 1044
12 1046
Flowkit Commands and Attributes 1046
check_flow 1046
create_flow 1048
create_flow_step 1049
define_feature 1051
delete_flow_config 1053
edit_flow 1054
end_flow 1056
flow Category Attributes 1056
flow_branch 1057
flow_caller_data 1057
flow_db_directory 1058
flow_error_errorinfo 1058
flow_error_message 1058
flow_error_write_db 1058
flow_exclude_time_for_init_flow 1058
flow_exit_when_done 1058
flow_feature_values 1058
flow_features 1059
flow_footer_tcl 1059
flow_header_tcl 1059
flow_hier_path 1059
flow_history 1059
flow_log_directory 1059

flow_log_prefix_generator 1059
flow_mail_on_error 1060
flow_mail_to 1060
flow_metrics_file 1061
flow_metrics_snapshot_parent_uuid 1061
flow_metrics_snapshot_uuid 1061
flow_overwrite_db 1061
flow_plugin_names 1061
flow_plugin_steps 1061
flow_post_db_overwrite 1061
flow_remark 1062
flow_report_directory 1062
flow_run_tag 1062
flow_schedule 1062
flow_starting_db 1062
flow_startup_directory 1062
flow_status_file 1062
flow_step_canonical_current 1063
flow_step_check_tcl 1063
flow_step_current 1063
flow_step_last 1063
flow_step_last_msg 1063
flow_step_last_status 1063
flow_step_next 1063
flow_summary_tcl 1064
flow_template_feature_definition 1064
flow_template_tools 1064
flow_template_type 1064
flow_template_version 1064
flow_user_templates 1064
flow_top 1064
flow_verbose 1065
flow_working_directory 1065
flow_yamllint_exec 1065
flowtool Category Attributes 1066
flowtool_exit_timeout 1066
flowtool_extra_arguments 1066

flowtool_metrics_qor_html 1066
flowtool_metrics_qor_excel 1066
flowtool_metrics_qor_text 1067
flowtool_metrics_qor_vivid 1067
flowtool_predict_full_names 1067
flowtool_summary_tcl 1067
get_feature 1067
get_flow_config 1068
init_flow 1070
is_feature 1070
is_flow 1072
read_flow 1073
report_flow 1074
reset_flow 1075
run_flow 1076
schedule_flow 1078
set_feature 1079
set_flow_config 1080
set_flowkit_read_db_args 1081
set_flowkit_write_db_args 1083
write_flow 1086
write_flow_template 1087
13 1092
General Commands and Attributes 1092
alias 1094
check_ccr 1097
check_netlist 1099
check_syntax 1103
compare_release 1105
convert_legacy_to_common_ui 1108
decrypt 1109
define_proc 1110
define_proc_arguments 1119
define_variables 1123
delete_floating_constants 1125
delete_dangling_ports 1127

deselect_obj 1128
encrypt 1129
eval_legacy 1130
get_all_layers 1132
get_common_ui_map 1134
get_compress_level 1136
get_layer_preference 1137
get_legacy 1140
get_message 1142
get_preference 1144
gui_bind_key 1145
gui_get_record_obj_by_info 1147
gui_move_cursor 1148
help 1149
is_common_ui_mode 1153
lminus 1154
man 1155
parse_proc_arguments 1157
print_message 1160
program Category Attributes 1162
- cmd_file 1162
- log_file 1163
- logv_file 1163
- program_major_version 1163
- program_name 1164
- program_short_name 1164
- program_version 1164
- selected 1165

puts 1165
redirect 1167
rename_obj 1170
report_area 1174
report_dangling_ports 1179
report_gate_count 1181
report_hidden_usage 1184
report_messages 1186
report_netlist_statistics 1189

report_obj 1190
report_obj Category Attributes 1195
  report_obj_display_limit 1195
report_summary 1195
resume 1202
select_obj 1203
set_compress_level 1205
set_current_design 1207
set_layer_preference 1208
set_license_check 1211
set_limited_access_feature 1214
set_log_post_cmd 1215
set_message 1216
set_preference 1219
set_proc_verbose 1235
set_window_preference 1238
source 1240
suspend 1241
system Category Attributes 1242
  auto_file_dir 1243
  auto_file_prefix 1243
  distributed_mmmc_disable_reports_auto_redirection 1244
  gui_verbose 1244
  layer_name_no_abbreviation 1244
  log_verbose_type 1245
  print_full_message_summary 1246
  read_db_directory 1246
  read_db_file_check 1246
  read_db_stop_at_design_in_memory 1246
  read_db_tool_name 1247
  read_db_version 1247
  script_search_path 1247
  source_echo_filename 1247
  set_db_verbose 1248
  soft_stack_size_limit 1248
  source_verbose 1248
  source_continue_on_error 1249

log_verbose_style 1251
source_verbose_line_length_limit 1252
tcl_error_info 1252
tcl_partial_cmd_argument_matching 1253
tcl_return_display_length_limit 1253
ui_precision 1254
ui_precision_capacitance 1254
ui_precision_derating 1254
ui_precision_power 1255
ui_precision_sensitivities 1255
ui_precision_timing 1255
write_db_auto_save_user_globals 1255
write_db_cmd_file_limit 1256
uniquify_filename 1257
update_names 1258
view_log 1267
gui_get_legend 1268
gui_set_legend 1270
read_taf 1273
report_disk 1274
time_info 1275
14 1278
GUI Commands 1278
clear_all_rulers 1281
create_gui_shape 1282
create_gui_text 1286
create_ruler 1290
delete_workspace 1292
edit_copy 1294
gui_add_marker 1301
gui_add_ui 1303
gui_attach_to_cursor 1309
gui_browse_db 1311
gui_clear_highlight 1313
gui_clear_hinst_colors 1316
gui_clear_trace_flightline 1318

gui_close_cell_view 1319
gui_create_floorplan_snapshot 1320
gui_create_user_bundle_net 1322
gui_delete_objs 1325
gui_delete_ui 1326
gui_delete_user_bundle_net 1328
gui_deselect 1330
gui_dim_foreground 1332
gui_find_ui 1333
gui_fit 1339
gui_get_box 1340
gui_get_coord 1341
gui_get_coords 1342
gui_get_ui 1343
gui_get_visible_bumps 1350
gui_get_visible_net_groups 1351
gui_get_visible_nets 1352
gui_group_hinst 1353
gui_hide 1355
gui_highlight 1356
gui_highlight_pin 1360
gui_highlight_pin_connection 1363
gui_list_marker 1367
gui_open_cell_view 1368
gui_open_schematic 1369
gui_pan 1375
gui_pan_center 1377
gui_pan_page 1378
gui_redo 1380
gui_redraw 1383
gui_register_edit_callback 1384
gui_remove_marker 1386
gui_report_trace_flightline 1387
gui_schematic_load_view 1390
gui_schematic_save_view 1391
gui_select 1392
gui_select_highlighted 1395

gui_set_obj_color 1396
gui_set_power_rail_display 1399
gui_set_power_rail_layers_nets 1411
gui_set_select_hinst_color 1418
gui_set_tool 1420
gui_set_ui 1422
gui_set_visible_bumps 1429
gui_set_visible_net_groups 1431
gui_set_visible_nets 1432
gui_show 1433
gui_show_edge_number 1434
gui_show_floorplan_snapshot 1436
gui_show_user_bundle_nets 1438
gui_trace_flightline 1441
gui_undo 1444
gui_ungroup_hinst 1447
gui_unregister_edit_callback 1450
gui_view_box 1452
gui_write_flightline 1453
gui_write_flow_gifs 1455
gui_write_picture 1461
gui_zoom 1465
read_power_rail_results 1468
read_workspace 1476
report_gui_edit_callback 1478
report_power_rail_results 1482
set_default_workspace 1492
view_last 1494
view_next 1495
write_color_preference 1496
write_hinst_colors 1498
write_to_gif 1500
write_workspace 1501
15 1504
ILM and FlexILM Commands and Attributes 1504
commit_module_model 1505

create_ilm_from_files 1507
flatten_ilm 1511
get_ilm_type 1512
ilm Category Attributes 1513
ilm_filter_internal_path 1513
ilm_keep_async 1514
ilm_keep_flatten 1514
ilm_keep_high_fanout_ports 1514
ilm_keep_inst_in_sdc 1515
ilm_keep_loopback 1515
ilm_max_num_insts 1515
ilm_max_num_registers 1516
ilm_slack_driven 1516
opt_cloud_design 1517
place_cloud_design 1520
read_ilm 1522
report_ilm_status 1525
reset_ilm 1527
set_ilm_type 1528
unflatten_ilm 1531
update_partition 1532
write_ilm 1534
read_ilm_eco_db 1538
write_ilm_eco_db 1540
16 1543
Import and Export Commands and Attributes 1543
add_route_via_defs 1544
add_route_vias Category Attributes 1547
add_route_vias_auto 1547
add_route_vias_ndr_only 1547
check_design 1548
check_unique 1553
compare_def 1554
create_pin_text 1559
create_tracks 1562
delete_tracks 1564

design Category Attributes 1565
design_backside_bottom_routing_layer 1565
design_backside_top_routing_layer 1566
design_bottom_routing_layer 1566
design_compressed_pg_db 1566
design_cong_effort 1566
design_dual_rail_via_pitch 1567
design_early_clock_flow 1567
design_express_route 1567
design_high_freq_interposer_flow 1567
design_ignore_followpin_vias 1568
design_merge_trim_shapes 1568
design_power_effort 1568
design_flow_effort 1569
design_process_node 1569
design_pessimistic_mode 1569
design_slack_weighting_method 1570
design_tech_node 1570
design_top_routing_layer 1570
design_trim_grid_group 1570
disconnect_dangling_port 1572
Import and Export Without Category Attributes 1574
ignore_max_via_stack_rule 1574
read_write_lef_check_uncolored_shapes 1574
write_def_stream_check_uncolored 1575
write_lec_files 1575
write_lec_directory_naming_style 1576
write_lec_dft_constraints 1576
write_lef_abstract_customer_header 1576
init_design 1577
init Category Attributes 1580
init_check_netlist 1581
init_check_output_pin_constant 1581
init_delete_floating_hnets 1582
init_design_netlist_type 1582
init_design_uniquify 1582
init_ground_nets 1583

init_ignore_pg_pin_polarity_check 1583
init_keep_empty_modules 1584
init_lef_files 1584
init_min_dbu_per_micron 1584
init_mmmc_files 1585
init_no_new_assigns 1585
init_oa_abstract_views 1586
init_oa_default_rule 1586
init_oa_design_cell 1587
init_oa_design_lib 1587
init_oa_design_view 1588
init_oa_foundry_rule 1588
init_oa_layout_views 1589
init_oa_ref_libs 1589
init_oa_search_libs 1590
init_oa_special_rule 1591
init_oa_tech_lib 1591
init_power_intent_files 1592
init_power_nets 1592
init_read_netlist_allow_port_mismatch 1592
init_read_netlist_allow_undefined_cells 1593
init_read_netlist_files 1593
init_sync_relative_path 1593
init_timing_enabled 1594
read_db 1594
read_def 1599
read_def Category Attributes 1604
read_def_check_mask_shifts 1604
read_mmmc 1605
read_name_mapping 1608
read_netlist 1610
read_physical 1613
read_physical Category Attributes 1618
read_physical_check_antenna 1618
read_physical_check_colored_shape 1618
read_physical_check_mask_shifts 1619
read_physical_extend_cell_obs_shapes_under_trim 1619

read_physical_merge_multi_port_to_single_port 1619
read_physical_must_join_all_ports 1619
read_physical_extend_polygon_cell_shapes 1620
read_stream 1620
relink_db_files 1624
reset_design 1626
run_abstract 1627
run_abstract Category Attributes 1629
run_abstract_abstract_blockage_cut_around_pin 1630
run_abstract_antenna_connectivity 1630
run_abstract_antenna_diffusion_geometry 1630
run_abstract_antenna_gate_geometry 1631
run_abstract_blockage_detailed_layers 1631
run_abstract_boundary_layers 1631
run_abstract_cell_symmetry 1631
run_abstract_export_lef_version 1632
run_abstract_extract_layers_power 1632
run_abstract_extract_layers_signal 1632
run_abstract_extract_pin_layers_power 1632
run_abstract_extract_pin_layers_signal 1633
run_abstract_input_cell_type 1633
run_abstract_input_stream_layer_map_file 1633
run_abstract_input_lef_tech_file 1634
run_abstract_keep_output_files 1634
run_abstract_pins_analog_names 1634
run_abstract_pins_clock_names 1634
run_abstract_pins_ground_names 1635
run_abstract_pins_output_names 1635
run_abstract_pins_power_names 1635
run_abstract_pins_text_pin_map 1635
run_abstract_pre_abstract_script 1635
run_abstract_selected_cells 1636
run_abstract_site_name 1636
run_abstract_verbose 1636
set_cell_power_domain 1637
set_library_unit 1639
set_power_lib_files 1641

trim_design 1643
update_lef_macro 1647
write_db 1648
write_db Category Attributes 1662
write_db_binary_timing_constraints 1662
write_db_create_read_file 1662
write_db_include_metal_fill_rules 1663
write_db_copy_timing_constraints_always 1663
write_db_save_unused_lef_block_names 1664
write_def 1665
write_def_by_section 1673
write_def Category Attributes 1680
write_def_hierarchy_delimiter 1680
write_def_compress_vias 1681
write_def_include_lef_ndr 1681
write_def_include_lef_vias 1681
write_def_polygon_die_area 1682
write_def_lef_out_version 1682
write_design 1683
write_do_lec 1686
write_lef_abstract 1689
write_mmmc 1699
write_name_mapping 1700
write_netlist 1702
write_netlist Category Attributes 1712
write_netlist_full_pin_out 1712
write_netlist_port_association_style 1712
write_stream 1713
write_stream Category Attributes 1724
write_stream_allow_path_type4 1725
write_stream_cell_instance_color 1725
write_stream_cell_master_color 1725
write_stream_cell_name_prefix 1725
write_stream_cell_name_suffix 1726
write_stream_check_map_file 1726
write_stream_compatible 1726
write_stream_define_via_name 1726

write_stream_ignore_fixed_mask 1727
write_stream_label_all_pin_shape 1727
write_stream_oasis_compression 1728
write_stream_oasis_layer_name 1728
write_stream_oasis_cell_offset 1728
write_stream_merge_append 1728
write_stream_pin_text_orientation 1728
write_stream_remove_nets 1729
write_stream_snap_to_mfg 1729
write_stream_stream_convert_rect_to_path 1729
write_stream_stream_version 1730
write_stream_text_size 1730
write_stream_uniquify_cell_names_prefix 1730
write_stream_via_names 1731
write_stream_virtual_connection 1731
write_via_defs 1731
report_gates 1734
report_sequential 1736
check_design Category Attributes 1738
check_design_across_hierarchy 1738
check_design_report_async_pins 1738
check_design_report_scan_pins 1738
check_design_threshold_fanout 1739
17 1740
Interactive ECO Commands and Attributes 1740
add_io_buffers 1742
add_notch_fill 1748
connect 1750
connect_hpin 1753
connect_pin 1755
create_diode 1758
create_hinst 1760
create_hnet 1762
create_hport 1765
create_inst 1767
create_module 1770

create_net 1771
create_port 1774
delete_assigns 1775
delete_bus_net 1780
delete_empty_hinsts 1782
delete_feedthru_buffer 1784
delete_hinst 1787
delete_hpin 1789
delete_inst 1791
delete_nets 1793
delete_notch_fill 1795
disconnect 1796
disconnect_hpin 1798
disconnect_pin 1799
eco_add_repeater 1801
eco_compare_netlist 1806
eco_delete_repeater 1809
eco_design 1811
eco_oa_design 1816
eco_read_def 1819
eco_remove_tied_inputs 1823
eco_swap_spare_cell 1825
eco_update_cell 1828
eco Category Attributes 1831
- eco_add_ports 1831
- eco_batch_mode 1832
- eco_delay_calculation_effort 1832
- eco_disable_change_net_name_for_flat_netlist 1833
- eco_honor_dont_touch 1833
- eco_honor_dont_use 1833
- eco_honor_fixed_status 1834
- eco_honor_fixed_wires 1834
- eco_honor_power_intent 1834
- eco_inherit_net_attribute 1835
- eco_check_logical_equivalence 1835
- eco_prefix 1836
- eco_preserve_hpin_function 1836

eco_refine_place 1836
eco_spread_inverter 1837
eco_update_only_layers 1837
eco_update_timing 1837
ECO Without Category Attributes 1838
bit_blasted_port_style 1838
group_instance_suffix 1839
end_eco 1840
fix_multi_drivers 1841
group 1842
gui_show_buffer_tree 1843
init_eco 1844
place_eco 1845
read_eco 1849
ungroup 1852
update_inst 1855
bitblast_ports 1858
delete_unintantiated_modules 1861
18 1863
Low Power Commands and Attributes 1863
add_power_switches 1864
Column Type Insertion Parameters 1868
Power Switch Prototyping Parameters 1868
Ring Type Insertion Parameters 1868
Syntax Conventions 1912
Syntax Examples 1912
check_power_domains 1914
check_power_intent 1917
check_power_switches 1918
clone_msv_gate 1921
commit_power_intent 1923
compare_power_intent 1926
cut_box_list_from_power_domain 1928
cut_power_domain_by_overlaps 1929
delete_power_intent 1930
delete_power_switches 1931

get_cpf_user_attributes 1933
get_pg_pins_use_signal_route 1934
get_upf_port_attr 1935
get_valid_power_switch_location 1936
identify_physical_power_domains 1937
move_power_switches 1938
power_intent Category Attributes 1941
power_intent_allow_power_domain_min_gap_zero 1941
power_intent_allow_shifter_voltage_mismatch 1941
power_intent_honor_power_domain_for_domain_crossing_route 1941
power_intent_honor_power_domain_for_intra_domain_route 1942
power_intent_do_not_use_top_domain_for_port_voltage 1942
power_intent_allow_back_to_back_isolation 1942
power_intent_allow_nested_default_domain 1942
power_intent_assume_iso_enable_pin_is_always_on 1943
power_intent_check_all_nets_for_domain_crossing 1943
power_intent_include_dot_lib_related_pg_pin 1943
power_intent_share_well_always_on_buffering_support 1943
power_intent_use_cpf_global_connect_for_always_on_buffer 1944
power_intent_use_cpf_global_connect_for_shifter 1944
power_intent_use_effective_domain_for_iso_shifter_insertion 1944
read_power_intent 1944
read_power_switches 1946
PSO File Generation and Usage 1946
read_power_switches_for_floorplan 1948
reorder_power_switch_chains 1949
report_isolation 1954
report_partition_boundary_ports 1955
report_power_domains 1956
report_power_switches 1960
report_shifters 1961
report_voltage 1963
route_secondary_pg_pins 1964
set_pg_pins_use_signal_route 1966
swap_cell_with_always_on_cell 1966
update_power_domain 1968
update_power_switches 1971

write_power_intent 1973
write_power_switches 1975
PSO File Generation and Usage 1975
write_power_switches_for_floorplan 1977
19 1978
Machine Learning Commands 1978
init_ml 1978
write_ml_design 1981
write_ml_library 1984
write_ml_timing_graph 1986
write_ml_timing_path 1988
20 1992
Metal and Via Fill Commands 1992
add_metal_fill 1992
add_metal_fill_signoff 2002
add_pg_fill 2012
Pegasus PG Fill 2012
add_track_fill 2015
add_via_fill 2018
delete_metal_fill 2020
fix_open_fill 2022
get_metal_fill_spacing_table 2023
get_pg_fill_config 2025
get_track_fill_config 2028
report_metal_fill 2030
set_metal_fill 2033
set_metal_fill_spacing_table 2039
set_pg_fill_config 2041
set_track_fill_config 2044
set_via_fill 2046
trim_metal_fill 2048
trim_metal_fill_near_net 2050
21 2053
Mixed Signal Commands 2053
delete_integration_route_constraints 2053

get_integration_route_constraint 2056
pull_block_constraints 2060
run_vsr 2063
set_integration_route_constraint 2067
write_integration_route_constraints 2080
22 2084
Multiple-CPU Processing Commands 2084
check_multi_cpu_usage 2084
get_distributed_hosts 2087
get_multi_cpu_usage 2089
monitor_distributed_hosts 2091
report_resource 2093
set_distributed_hosts 2097
set_multi_cpu_usage 2104
23 2112
OpenAccess Commands and Attributes 2112
compare_oa_cellview 2112
create_oa_lib 2115
create_oa_lib_property 2117
get_oa_default_rule_lib 2119
get_oa_design_data 2121
oa_read_write Category Attributes 2123
oa_allow_analysis_only 2124
oa_allow_bit_connection 2124
oa_allow_tech_update 2124
oa_bindkey_file 2125
oa_cell_view_dir 2125
oa_convert_diagonal_path_to 2125
oa_cut_rows 2126
oa_display_resource_file 2126
oa_display_resource_file_in_library 2126
oa_drc_fill_purpose 2126
oa_enclose_quick_abstract_pins 2127
oa_full_layer_list 2127
oa_full_path 2127
oa_inst_placed_if_none 2127

oa_lib_create_mode 2128
oa_lock 2128
oa_logic_only_import 2129
oa_merge_trim 2129
oa_new_lib_compress_level 2130
oa_pin_purpose 2130
oa_push_pin_constraint 2130
oa_quick_abstract_for_custom_cells 2130
oa_read_system_reserved_layers 2131
oa_silently_ignore_unsupported_vias 2131
oa_text_purpose 2131
oa_tie_net 2131
oa_update_mode 2132
oa_use_virtuoso_bindkey 2133
oa_use_virtuoso_color 2133
oa_view_sub_type 2133
oa_write_cds_fixed_vias 2133
oa_write_mask_data_locked 2134
oa_write_net_voltage 2134
oa_write_relative_path 2134
read_oa 2135
register_trigger 2138
report_oa_lib 2139
set_oa_cell_binding 2141
unlock_oa_design 2142
update_oa_lib 2143
write_lef_library 2145
write_oa 2147
write_oa_abstract 2150
write_oa_black_boxes 2152
write_oa_techfile 2154
24 2156
Partition Commands and Attributes 2156
add_connection_to_neighbor_partition 2159
add_partition_feedthrus 2160
align_partition_clones 2164

align_pins 2167
assemble_design 2170
assign_io_pins 2174
assign_partition_pins 2178
assign_pins Category Attributes 2184
- assign_pins_advanced_node_rule_support 2184
- assign_pins_allow_non_ndr_net_pins_on_ndr_tracks 2184
- assign_pins_allow_unconnected_in_abutted_edge 2185
- assign_pins_blocked_boundary_macro_distance 2185
- assign_pins_edit_in_batch 2185
- assign_pins_force_abutment_with_fixed 2186
- assign_pins_max_channel_width_as_abutted 2186
- assign_pins_max_distance_pair 2187
- assign_pins_off_stripe 2187
- assign_pins_pin_to_stripe_distance 2187
- assign_pins_pin_to_via_distance_non_preferred_direction 2188
- assign_pins_pin_to_via_distance_preferred_direction 2188
- assign_pins_promoted_macro_bottom_layer 2188
- assign_pins_promoted_macro_top_layer 2189
- assign_pins_restricted_boundary_macro_distance 2189
- assign_pins_strict_abutment 2189

change_current_design 2191
check_bus_guides 2192
check_hier_route 2194
check_pin_assignment 2196
clone_partition_obj 2201
commit_partitions 2203
commit_pushdown_eco 2208
compare_pin_assignment_statistics 2209
connect_macro_feedthru 2210
convert_blackbox_to_fence 2213
convert_fence_to_black_box 2214
create_blackbox 2215
create_net_group 2219
create_partition 2221
create_partition_feedthru 2226
create_partition_wrapper 2227

create_pin_blockage 2229
create_pin_group 2231
create_pin_guide 2233
create_pushdown_eco 2235
delete_all_partition_cuts 2236
delete_all_partition_feedthrus 2237
delete_black_boxes 2238
delete_net_group 2239
delete_nets_from_net_group 2240
delete_partition_cuts 2241
delete_partitions 2242
delete_pin_blockages 2244
delete_pin_from_pin_group 2245
delete_pin_groups 2246
delete_pin_guides 2247
edit_pin 2249
fix_feedthru_assign 2257
flatten_cover_cell 2258
flatten_partitions 2259
get_black_box_area 2261
get_clone_partition_orient 2262
get_partition_pin_status 2263
get_pin_constraint 2264
gui_highlight_feedthru_nets 2266
gui_show_partition_wire_cross_over 2267
highlight_sync_obj 2270
legalize_pins 2271
partition_floorplan Category Attributes 2274
  partition_floorplan_export 2274
  partition_floorplan_import 2274
partition_push_network 2275
partition_push_special_wires 2278
place_cells_at_center_for_feedthrus 2280
place_partition_clone 2281
read_black_box_netlist 2283
read_hierarchical_def 2284
read_partition_pins 2287

report_partitions 2289
report_pin_assignment_statistics 2291
report_two_pin_chain 2293
report_unaligned_nets 2295
reset_partition_pins 2300
reset_pin_constraint 2301
route_hierarchical_clock 2303
set_blackbox_master_orient 2305
set_blackbox_master_to_r0 2306
set_clone_partition_orient 2307
set_partition_area_io 2308
set_partition_pin_status 2309
set_partition_pin_use 2311
set_pin_constraint 2312
set_promoted_macro_pin 2316
undo_uniquify_partition 2318
uniquify_partition 2320
update_net_group 2321
update_partition_cell_obs 2322
update_partition_cut 2323
update_pin_group 2324
write_black_box_lef 2325
write_generalized_feedthru_paths 2326
write_net_groups_topo 2328
write_partition_pins 2329
write_partitions 2330
25 2334
Placement Commands and Attributes 2334
add_decomp_filler 2338
add_endcaps 2341
add_endcaps Category Attributes 2344
add_endcaps_avoid_two_sites_cell_abut 2346
add_endcaps_bottom_edge 2347
add_endcaps_boundary_tap 2347
add_endcaps_boundary_tap_swap_flow 2347
add_endcaps_cells 2347

add_endcaps_create_rows 2347
add_endcaps_double_height_incorner_protrusion 2348
add_endcaps_flip_y 2348
add_endcaps_incremental_left_edge 2348
add_endcaps_incremental_right_edge 2348
add_endcaps_left_bottom_corner 2349
add_endcaps_left_bottom_corner_even 2349
add_endcaps_left_bottom_corner_neighbor 2349
add_endcaps_left_bottom_corner_odd 2349
add_endcaps_left_bottom_edge 2350
add_endcaps_left_bottom_edge_even 2350
add_endcaps_left_bottom_edge_neighbor 2350
add_endcaps_left_bottom_edge_odd 2350
add_endcaps_left_edge 2351
add_endcaps_left_edge_bottom_border 2351
add_endcaps_left_edge_even 2351
add_endcaps_left_edge_odd 2351
add_endcaps_left_edge_top_border 2351
add_endcaps_left_top_corner 2352
add_endcaps_left_top_corner_even 2352
add_endcaps_left_top_corner_neighbor 2352
add_endcaps_left_top_corner_odd 2352
add_endcaps_left_top_edge 2353
add_endcaps_right_corner_bottom_border 2353
add_endcaps_right_corner_top_border 2353
add_endcaps_left_top_edge_even 2353
add_endcaps_left_top_edge_neighbor 2353
add_endcaps_left_top_edge_odd 2354
add_endcaps_min_horizontal_channel_width 2354
add_endcaps_min_jog_height 2354
add_endcaps_min_jog_width 2354
add_endcaps_min_vertical_channel_width 2355
add_endcaps_prefix 2355
add_endcaps_right_bottom_corner 2355
add_endcaps_right_bottom_corner_even 2355
add_endcaps_right_bottom_corner_neighbor 2355
add_endcaps_right_bottom_corner_odd 2356

add_endcaps_right_bottom_edge 2356
add_endcaps_right_bottom_edge_even 2356
add_endcaps_right_bottom_edge_neighbor 2356
add_endcaps_right_bottom_edge_odd 2357
add_endcaps_right_edge 2357
add_endcaps_right_edge_bottom_border 2357
add_endcaps_right_edge_even 2357
add_endcaps_right_edge_odd 2358
add_endcaps_right_edge_top_border 2358
add_endcaps_right_top_corner 2358
add_endcaps_right_top_corner_even 2358
add_endcaps_right_top_corner_neighbor 2358
add_endcaps_right_top_corner_odd 2359
add_endcaps_right_top_edge 2359
add_endcaps_right_top_edge_even 2359
add_endcaps_right_top_edge_neighbor 2359
add_endcaps_right_top_edge_odd 2360
add_endcaps_left_corner_bottom_border 2360
add_endcaps_left_corner_top_border 2360
add_endcaps_top_bottom_edge 2360
add_endcaps_top_edge 2360
add_endcaps_use_even_odd_sites 2361
add_endcaps_wall_keepout_from_vertical_boundary 2361
add_endcaps_wall_offset 2361
add_endcaps_wall_pitch 2361
add_endcaps_wall_to_convex_corner_spacing 2362
add_filler_gaps 2363
add_fillers 2366
add_fillers Category Attributes 2375
add_fillers_cells 2376
add_fillers_avoid_abutment_patterns 2376
add_fillers_cell_name_style 2376
add_fillers_check_different_cells 2377
add_fillers_check_drc 2377
add_fillers_create_rows 2377
add_fillers_distribute_implant_evenly 2377
add_fillers_eco_mode 2378

add_fillers_horizontal_exception_cell 2378
add_fillers_honor_preroute_as_obs 2378
add_fillers_keep_fixed 2379
add_fillers_horizontal_max_length 2379
add_fillers_horizontal_repair_cell 2379
add_fillers_no_single_site_gap 2379
add_fillers_prefix 2379
add_fillers_preserve_user_order 2380
add_fillers_scheme 2380
add_fillers_swap_cell 2380
add_fillers_vertical_stack_exception_cell 2381
add_fillers_vertical_stack_left_edge_exception_cell 2381
add_fillers_vertical_stack_max_length 2381
add_fillers_vertical_stack_repair_cell 2381
add_fillers_vertical_stack_repair_edge 2381
add_fillers_vertical_stack_right_edge_exception_cell 2382
add_fillers_with_drc 2382
add_fillers_y_flip_type 2382
add_gate_array_filler 2383
add_mim_cap 2386
add_spare_insts 2388
add_tap_walls 2391
add_well_taps 2393
add_well_taps Category Attributes 2400
add_well_taps_avoid_abutment 2401
add_well_taps_avoid_vertical_well_abutment 2401
add_well_taps_block_boundary_only 2401
add_well_taps_bottom_tap_cell 2401
add_well_taps_bottom_termination_cell 2402
add_well_taps_cell 2402
add_well_taps_channel_offset 2402
add_well_taps_check_channel 2402
add_well_taps_column_cells 2403
add_well_taps_create_rows 2403
add_well_taps_disable_check_zone_at_boundary 2403
add_well_taps_in_row_offset 2403
add_well_taps_insert_cells 2403

add_well_taps_rule 2404
add_well_taps_site_offset 2405
add_well_taps_tap_function_cells 2405
add_well_taps_tap_termination_alignment 2405
add_well_taps_termination_align 2405
add_well_taps_termination_cells 2406
add_well_taps_top_tap_cell 2406
add_well_taps_vertical_boundary_spacing 2406
add_well_taps_top_termination_cell 2407
add_well_taps_well_cut_cells 2407
check_filler 2408
check_library 2411
check_place 2413
create_gate_array_filler_group 2418
create_inst_space_group 2420
create_pg_keep_out 2423
create_pipeline_bus_guide 2426
create_pipeline_net_group 2428
create_scan_chain 2431
create_scan_chain_partition 2433
create_spare_module 2435
delete_all_cell_padding 2437
delete_all_scan_cells 2438
delete_base_cell_edge_type 2439
delete_cell_edge_spacing 2440
delete_cell_padding 2441
delete_cell_stack_area 2443
delete_cell_stack_group 2444
delete_cell_virtual_align 2445
delete_filler 2446
delete_gate_array_filler_group 2449
delete_inst_padding 2450
delete_inst_space_group 2452
delete_mim_cap 2453
delete_pg_keep_out 2454
delete_pipeline_net_group 2456
delete_scan_cell 2457

delete_scan_chain 2458
delete_scan_chain_partition 2460
delete_spare_module 2462
gui_clear_scan_chains 2464
gui_clear_spare_cells 2465
gui_show_scan_chains 2466
gui_show_spare_cells 2467
place_clones 2468
place_connected 2469
place_design 2473
place_detail 2476
place_fix_congestion 2479
place_inst 2480
place_jtag 2483
place_opt_design 2488
place_pad 2491
place_pipelines 2493
place_spare_modules 2495
place Category Attributes 2499
- place_cell_edge_spacing 2501
- place_design_floorplan_mode 2502
- place_design_refine_macro 2502
- place_design_refine_place 2502
- place_detail_activity_power_driven 2503
- place_detail_allow_border_pin_abut 2503
- place_detail_allow_single_height_row_symmetry_x 2503
- place_detail_check_cut_spacing 2503
- place_detail_check_inst_space_group 2504
- place_detail_check_route 2504
- place_detail_eco_max_distance 2505
- place_detail_eco_priority_insts 2505
- place_detail_fixed_shifter 2505
- place_detail_honor_inst_pad 2506
- place_detail_irdrop_aware_effort 2506
- place_detail_io_pin_blockage 2506
- place_detail_ir_aware_max_drive_strength 2507
- place_detail_irdrop_aware_timing_effort 2507

place_detail_irdrop_region_number 2507
place_detail_legalization_inst_gap 2508
place_detail_m3_stripe_push_down 2508
place_detail_m3_stripe_shrink 2508
place_detail_max_shifter_depth 2509
place_detail_max_shifter_column_depth 2509
place_detail_max_shifter_row_depth 2509
place_detail_no_filler_without_implant 2509
place_detail_use_gate_array_filler_groups 2510
place_detail_use_no_diffusion_one_site_filler 2510
place_detail_context_aware_legal 2510
place_detail_color_aware_legal 2510
place_detail_pad_fixed_insts 2511
place_detail_pad_physical_cells 2511
place_detail_preroute_as_obs 2511
place_detail_preserve_routing 2511
place_detail_remove_affected_routing 2512
place_detail_sdp_alignment_in_refine 2512
place_detail_swap_eeq_cells 2512
place_detail_use_check_drc 2512
place_detail_use_diffusion_transition_fill 2513
place_detail_wire_length_opt_effort 2513
place_global_activity_power_driven_effort 2513
place_global_align_macro 2513
place_global_allow_3d_stack 2514
place_global_clock_power_driven_effort 2514
place_global_activity_power_driven 2514
place_global_auto_blockage_in_channel 2514
place_global_clock_gate_aware 2515
place_global_clock_power_driven 2515
place_global_cong_effort 2515
place_global_cpg_effort 2516
place_global_cpg_file 2516
place_global_enable_advanced_pipeline 2516
place_global_enable_distributed_place 2517
place_global_ignore_scan 2517
place_global_ignore_spare 2517

place_global_max_density 2518
place_global_module_aware_spare 2518
place_global_module_padding 2519
place_global_place_io_pins 2519
place_global_sdp_alignment 2520
place_global_sdp_place 2520
place_global_reorder_scan 2520
place_global_soft_guide_strength 2520
place_global_timing_effort 2521
place_global_uniform_density 2521
place_hard_fence 2521
place_opt_post_place_tcl 2522
place_opt_run_global_place 2522
place_spare_update_timing_graph 2522
place Category Object Attributes 2523
reorder_scan 2524
reorder_scan Category Attributes 2528
reorder_scan_add_scan_port_prefix 2528
reorder_scan_clock_aware 2529
reorder_scan_comp_logic 2529
reorder_scan_enable_for_partition 2529
reorder_scan_keep_hport 2530
reorder_scan_keep_hinst_port_name 2530
reorder_scan_prefer_horizontal 2530
reorder_scan_prefer_vertical 2530
reorder_scan_effort 2531
reorder_scan_skip_mode 2531
reorder_scan_swap_effort 2532
report_cell_edge_spacing 2533
report_cell_edge_type 2534
report_cell_padding 2535
report_cell_stack_area 2537
report_cell_stack_group 2538
report_cell_virtual_align 2539
report_congestion 2540
report_density_in_area 2545
report_density_map 2547

report_inst_padding 2550
report_inst_space_group 2551
report_jtag_inst 2552
report_pg_keep_out 2553
report_pin_density 2554
report_pin_density_map 2556
report_pin_density Category Attributes 2559
  report_pin_density_map_grid_unit 2559
  report_pin_density_map_grid_size 2559
  report_pin_density_map_threshold 2560
  report_pin_density_map_display_step 2560
report_pipelines 2561
report_place_density Category Attributes 2563
  report_place_density_map_grid_size 2563
  report_place_density_map_grid_unit 2563
  report_place_density_map_ignore_block 2564
  report_place_density_map_ignore_filler 2564
  report_place_density_map_threshold 2564
report_scan_cells 2564
report_scan_chain 2566
report_scan_chain_partitions 2567
reset_jtag 2568
reset_selective_blockage_gate 2570
reset_spare_insts 2571
set_cell_edge_spacing 2573
set_cell_edge_type 2576
set_cell_padding 2580
set_cell_stack_area 2583
set_cell_stack_group 2585
set_cell_virtual_align 2586
set_inst_padding 2588
set_jtag 2590
set_jtag_to_unplaced 2593
set_lockup_element 2594
set_scan_cell 2596
set_selective_blockage_gate 2598
set_spare_insts 2599

swap_well_taps 2601
trace_jtag 2603
trace_scan 2605
update_inbound_cell_site 2607
update_pipeline_net_group 2609
update_vertical_cell_edge_constraint 2612
pack_align_macros 2614
26 2617
Power Calculation Commands and Attributes 2617
encrypt_thermal_file 2617
get_dynamic_power_simulation 2619
get_default_switching_activity 2620
power Category Attributes 2621
power_add_simulation 2625
power_adjust_input_activity_in_iterations 2625
power_adjust_macro_activity_in_iterations 2625
power_analysis_temperature 2625
power_annotation_detail_report 2626
power_auto_twf_delay_annotation 2626
power_average_rise_fall_cap 2626
power_block_independent_peakpower 2626
power_bulk_pins 2627
power_capacity 2627
power_case_insensitive_mapping 2627
power_clock_source_as_clock 2628
power_comprehensive_automapping 2628
power_constant_override 2628
power_corner 2628
power_create_driver_db 2629
power_current_generation_method 2629
power_db_name 2629
power_decap_cell_list 2629
power_default_frequency 2630
power_default_slew 2630
power_default_supply_voltage 2630
power_disable_clock_gate_clipping 2630

power_disable_leakage_scaling 2630
power_disable_static 2631
power_distribute_switching_power 2631
power_distributed_combine_report_format 2631
power_distributed_setup 2632
power_domain_based_clipping 2632
power_dynamic_glitch_filter 2632
power_enable_auto_queue 2633
power_enable_auto_mapping 2633
power_enable_disk_mapping 2633
power_enable_duty_propagation_with_global_activity 2634
power_enable_dynamic_current_slew_load_interpolation 2634
power_dynamic_scale_clock_by_frequency 2634
power_dynamic_scale_clock_by_name 2634
power_dynamic_vectorless_ranking_methods 2634
power_enable_dynamic_scaling 2635
power_enable_flop_state_propagation 2635
power_enable_generated_clock 2635
power_enable_input_net_power 2636
power_enable_interactive_reports 2636
power_enable_mt_state propagation 2636
power_enable_multithread_reports 2636
power_enable_power_target_flow 2637
power_enable_pba_for_tempus_pi 2637
power_enable_rtl_dynamic_vector_based 2637
power_enable_scan_report 2637
power_enable_slew_based_ccs_pin_cap 2638
power_enable_state_propagation 2638
power_enable_tempus_pi 2638
power_enable_xp 2639
power_enhanced_blackbox_avg 2639
power_enhanced_blackbox_max 2639
power_equivalent_annotation 2639
power_event_based_leakage_power 2639
power_external_load_config_file 2640
power_extraction_tech_file 2640
power_extractor_include 2640

power_fanout_limit 2641
power_force_library_merging 2641
power_from_x_transition_factor 2641
power_from_z_transition_factor 2641
power_generate_activity_mapping_report 2641
power_generate_current_for_rail 2642
power_generate_flop_ranking_data 2642
power_generate_static_report_from_state_propagation 2642
power_generate_leakage_power_map_based_on_calculated_leakage 2642
power_grid_libraries 2643
power_handle_glitch 2643
power_handle_tristate 2643
power_hier_delimiter 2643
power_honor_combinational_logic_on_clock_net 2644
power_honor_negative_energy 2644
power_honor_net_activity 2644
power_honor_non_mission_leakage 2644
power_hybrid_analysis 2645
power_ignore_control_signals 2645
power_ignore_data_phase_for_clock 2645
power_ignore_end_toggles_in_profile 2646
power_ignore_glitches_at_same_time_stamp 2646
power_ignore_inout_pin_cap 2646
power_include_file 2646
power_include_initial_x_transitions 2647
power_include_sequential_clock_pin_power 2647
power_include_timing_in_current_file 2647
power_ir_derated_timing_view 2647
power_keep_clock_gate_ratio_in_iterations 2648
power_leakage_scale_factor_for_temperature 2648
power_lib_files 2648
power_library_preference 2648
power_match_state_for_logic_x 2649
power_multibit_flop_toggle_behavior 2649
power_multi_scenario_simulation 2649
power_merge_switched_net_currents 2650
power_method 2650

power_min_leaf_count 2651
power_off_pg_nets 2651
power_output_current_data_prefix 2652
power_output_dir 2652
power_partition_count 2652
power_partition_twf 2652
power_pin_based_twf 2652
power_precision 2653
power_pre_simulation_empty_period 2653
power_pre_simulation_period 2653
power_pre_simulation_power_exclude_period 2653
power_quit_on_activity_coverage_threshold 2654
power_read_rcdb 2654
power_relax_arc_match 2654
power_report_black_boxes 2654
power_report_idle_instances 2655
power_report_instance_switching_info 2655
power_report_instance_switching_list 2656
power_report_library_usage 2656
power_report_missing_bulk_connectivity 2657
power_report_missing_input 2657
power_report_missing_nets 2657
power_report_scan_chain_statistics 2657
power_report_statistics 2658
power_report_time_display_fraction_digits 2658
power_report_twf_attributes 2658
power_reuse_flop_ranking_data 2659
power_reuse_flop_ranking_data_hier 2659
power_scale_to_sdc_clock_frequency 2660
power_scan_chain_activity 2660
power_scan_chain_name_pattern 2660
power_scan_control_file 2661
power_scan_multi_bit_flop_chain_type 2661
power_settling_buffer 2661
power_smart_window 2662
power_split_bus_power 2662
power_start_time_alignment 2662

power_state_dependent_leakage 2663
power_stateprop_ignore_unannot_pins 2663
power_static_multi_mode_scenario_file 2663
power_static_netlist 2663
power_switching_power_on_rise_only 2664
power_thermal_input_file 2664
power_thermal_leakage_temperature_scale_table_file 2665
power_to_x_transition_factor 2665
power_to_z_transition_factor 2665
power_transition_factor_based_duty 2665
power_transition_time_method 2666
power_twf_delay_annotation 2666
power_twf_load_cap 2666
power_use_cell_leakage_power_density 2667
power_use_fastest_clock_for_dynamic_scheduling 2667
power_use_lef_for_missing_cells 2667
power_use_physical_partition 2667
power_worst_case_vector_activity 2668
power_worst_step_size 2668
power_worst_window_count 2668
power_worst_window_coverage 2668
power_worst_window_reports 2669
power_worst_window_size 2669
power_worst_window_type 2669
power_use_zero_delay_vector_file 2670
power_vector_based_multithread 2670
power_vector_profile_mode 2670
power_write_bbox 2670
power_write_db 2671
power_write_default_pti_files 2671
power_write_dynamic_currents 2671
power_write_gui_db 2671
power_write_profiling_db 2672
power_write_simulation_db 2672
power_write_static_currents 2672
power_x_count_transition_using_3_states 2673
power_x_transition_factor 2673

power_z_transition_factor 2673
power_zero_delay_vector_toggle_shift 2673
propagate_activity 2674
query_power_data 2676
read_activity_file 2679
read_activity_map_file 2686
read_power_db 2689
report_inst_power 2691
report_power 2692
report_unannotated_nets 2709
reset_power_activity 2711
set_default_switching_activity 2712
set_dynamic_power_simulation 2716
set_instance_temperature_file 2718
set_power 2719
set_power_calculation_temperature 2727
set_power_include_file 2728
set_power_output_dir 2729
set_switching_activity 2730
set_twf_attribute 2734
set_virtual_clock_network_parameters 2738
write_power_constraints 2740
write_tcf 2741
27 2744
Power-Grid Library Commands 2744
check_pg_library 2744
check_pgv 2752
merge_pg_library 2757
rename_pg_library 2760
set_advanced_pg_library_mode 2761
set_pg_library_mode 2776
trim_pg_library 2792
write_pg_library 2794
write_tech_pg_lib 2796
write_xpgv 2798
28 2801

Power Planning Commands and Attributes 2801
add_power_mesh_colors 2801
add_reinforce_pg 2803
add_reinforce_pg Category Attributes 2811
add_reinforce_pg_auto_ir_fix_effort 2811
add_reinforce_pg_critical_path_slack 2811
add_reinforce_pg_irdrop_hierarchical_block 2811
add_reinforce_pg_irdrop_hierarchical_top_rail_analysis_directory 2812
add_reinforce_pg_respect_defined_nets 2812
add_reinforce_pg_respect_routes 2812
add_reinforce_pg_respect_stdcell_geometry 2813
add_reinforce_pg_timing_aware_effort 2813
add_rings 2814
add_rings Category Attributes 2820
add_rings_avoid_short 2820
add_rings_break_core_ring_io_list 2821
add_rings_continue_on_no_selection 2821
add_rings_detailed_log 2821
add_rings_extend_block_ring_search_distance 2821
add_rings_extend_core_ring_search_distance 2821
add_rings_extend_merge_with_pre_wires 2821
add_rings_extend_over_row 2822
add_rings_extend_search_nets 2822
add_rings_extend_stripe_search_distance 2822
add_rings_gap_width_without_io 2822
add_rings_ignore_rows 2822
add_rings_max_via_size 2822
add_rings_orthogonal_only 2823
add_rings_skip_shared_inner_ring 2823
add_rings_skip_via_on_pin 2823
add_rings_skip_via_on_wire_shape 2823
add_rings_spacing_from_block 2823
add_rings_split_long_via 2824
add_rings_stacked_via_bottom_layer 2824
add_rings_stacked_via_top_layer 2824
add_rings_target 2824

add_rings_via_using_exact_crossover_size 2824
add_rings_wire_center_offset 2825
add_split_power_vias 2825
add_stripes 2829
add_stripes Category Attributes 2842
add_stripes_allow_jog 2843
add_stripes_allow_non_preferred_dir 2844
add_stripes_area 2844
add_stripes_blocks_without_same_net 2844
add_stripes_break_at 2844
add_stripes_continue_on_no_selection 2846
add_stripes_detailed_log 2846
add_stripes_domain_offset_from_core 2846
add_stripes_extend_to_closest_target 2846
add_stripes_extend_to_first_ring 2846
add_stripes_ignore_block_check 2847
add_stripes_ignore_block_ring_during_break 2847
add_stripes_ignore_drc 2847
add_stripes_ignore_non_default_domains 2847
add_stripes_in_cell_allow_shift 2847
add_stripes_in_cell_only 2848
add_stripes_keep_pitch_after_snap 2848
add_stripes_mask_color_balance 2849
add_stripes_max_extension_distance 2850
add_stripes_max_via_size 2850
add_stripes_merge_with_all_layers 2851
add_stripes_mesh_via 2851
add_stripes_offset_from_core 2851
add_stripes_opt_stripe_for_routing_track 2852
add_stripes_orthogonal_offset 2853
add_stripes_orthogonal_only 2853
add_stripes_over_row_extension 2854
add_stripes_partial_set_through_domain 2854
add_stripes_remove_floating_stapling 2854
add_stripes_remove_floating_stripe_over_block 2854
add_stripes_remove_stripe_under_ring 2854
add_stripes_respect_routes 2854

add_stripes_route_over_rows_only 2855
add_stripes_rows_without_stripes_only 2855
add_stripes_same_size_stack_vias 2855
add_stripes_skip_via_on_pin 2855
add_stripes_skip_via_on_wire_shape 2856
add_stripes_spacing_from_block 2856
add_stripes_spacing_type 2856
add_stripes_split_long_via 2856
add_stripes_split_vias 2856
add_stripes_split_wire_spacing 2857
add_stripes_split_wire_weight 2857
add_stripes_split_wire_width 2857
add_stripes_stacked_via_bottom_layer 2857
add_stripes_stacked_via_top_layer 2857
add_stripes_stapling_extend_to_minimum_spacing 2857
add_stripes_stapling_nets_style 2858
add_stripes_stapling_shift 2860
add_stripes_stop_at_closest_target 2860
add_stripes_stop_at_last_wire_for_area 2860
add_stripes_stripe_min_length 2860
add_stripes_stripe_min_width 2860
add_stripes_switch_cell_name 2861
add_stripes_switch_layer_overlap_length 2861
add_stripes_trim_antenna_back_to_shape 2861
add_stripes_trim_antenna_max_distance 2862
add_stripes_trim_stripe 2862
add_stripes_use_exact_spacing 2862
add_stripes_use_point_to_point_route 2863
add_stripes_use_stripe_width 2863
add_stripes_via_using_exact_crossover_size 2863
add_stripes_width_file 2863
add_target_pg 2864
add_target_pg Category Attributes 2870
add_target_pg_allow_weak_connect 2870
add_target_pg_max_extension_distance 2870
add_target_pg_pins 2871
add_target_pg_pins_group_distance 2871

add_target_pg_respect_routes 2872
add_target_pg_share_resource 2872
generate_special_via Category Attributes 2873
generate_special_via_accuracy_effort 2874
generate_special_via_add_pin_to_pin_vias 2875
generate_special_via_align_merged_stack_via_metal 2875
generate_special_via_allow_via_expand 2875
generate_special_via_allow_wire_shape_change 2875
generate_special_via_area_only 2876
generate_special_via_check_signal_routes 2877
generate_special_via_create_double_row_cut_via 2877
generate_special_via_create_max_row_cut_via 2877
generate_special_via_cut_class_preference 2878
generate_special_via_disable_via_merge 2878
generate_special_via_enable_check_drc 2879
generate_special_via_extend_out_wire_end 2879
generate_special_via_full_cut_via_only 2880
generate_special_via_hookup_contact_max 2880
generate_special_via_hookup_contact_pg_track 2881
generate_special_via_hookup_fixed_grid 2881
generate_special_via_hookup_min_distance 2883
generate_special_via_hookup_rail_pair 2883
generate_special_via_hookup_via_distance 2883
generate_special_via_hookup_via_style 2884
generate_special_via_hookup_via_rule 2886
generate_special_via_hookup_virtual_trim_grid 2886
generate_special_via_ignore_design_boundary 2887
generate_special_via_ignore_drc 2887
generate_special_via_ignore_rule_enclosure 2888
generate_special_via_inherit_wire_status 2888
generate_special_via_keep_existing_via 2888
generate_special_via_keep_fixed_via 2888
generate_special_via_mincut_preference 2888
generate_special_via_opt_cross_via 2889
generate_special_via_opt_via_on_routing_track 2890
generate_special_via_parameterized_via_only 2890
generate_special_via_partial_overlap_threshold 2890

generate_special_via_preferred_vias_only 2891
generate_special_via_prefix 2891
generate_special_via_reference_boundary 2891
generate_special_via_respect_stdcell_geometry 2891
generate_special_via_respect_wire_ndr 2892
generate_special_via_rule_preference 2892
generate_special_via_set_via_expand_dir 2892
generate_special_via_snap_via_center_to_grid 2892
generate_special_via_split_long_via_global_grid 2894
generate_special_via_symmetrical_via_only 2895
generate_special_via_use_track_offset 2895
generate_special_via_use_trim_metal_enclosure 2895
trim_pg 2895
update_power_vias 2898
route_pg 2902
29 2906
Power Route Commands and Attributes 2906
fix_via 2906
report_power_route 2908
route_special 2910
route_special Category Attributes 2925
route_special_allow_non_preferred_direction_route 2926
route_special_avoid_over_core_row_layer 2926
route_special_block_pin_connect_ring_pin_corners 2927
route_special_block_pin_route_with_pin_width 2927
route_special_connect_broken_core_pin 2927
route_special_core_pin_ignore_obs 2929
route_special_core_pin_length 2929
route_special_core_pin_length_as_inst 2929
route_special_core_pin_max_via_scale 2930
route_special_core_pin_merge_limit 2930
route_special_core_pin_refer_to_follow_pin 2930
route_special_core_pin_reference_macro 2931
route_special_core_pin_site_rail_width 2931
route_special_core_pin_snap_to 2932
route_special_core_pin_stop_route 2933

route_special_endcap_as_core 2933
route_special_extend_nearest_target 2933
route_special_jog_threshold_ratio 2933
route_special_layer_preferred_direction_cost 2933
route_special_layer_non_preferred_direction_cost 2934
route_special_pad_pin_min_via_size 2934
route_special_pad_pin_split 2934
route_special_pad_ring_use_lef 2935
route_special_pg_pin_as_signal 2936
route_special_secondary_pin_max_gap 2936
route_special_secondary_pin_rail_width 2936
route_special_signal_pin_as_pg 2936
route_special_split_long_via 2937
route_special_target_number 2937
route_special_target_search_distance 2937
route_special_time_limit 2937
route_special_via_connect_to_shape 2938
route_special_via_through_to_closest_ring 2938
route_special_welltap_as_endcap 2938
30 2939
Prototyping Commands and Attributes 2939
FlexModel Commands and Attributes 2939
assemble_proto_model 2940
convert_proto_model 2943
create_flexfiller_route_blockage 2946
create_proto_model 2948
create_proto_net_delay_model 2951
get_proto_model 2952
gui_highlight_proto_models 2956
identify_proto_model 2960
proto_design 2963
proto_design Category Attributes 2965
proto Category Attributes 2968
report_proto_models 2983
set_proto_model 2988
set_proto_model_physical_constraint 2992

set_proto_timing_settings 2994
write_repeater_rules 2996
SoC Architecture Info (SAI) Commands 3000
add_sai_boundary_flops 3001
end_sai 3004
read_sai 3005
report_sai_constraint 3006
31 3011
Rail Analysis Commands 3011
add_what_if_shapes 3012
connect_die_package 3016
convert_stream_to_def 3018
generate_esd_rlrp_report 3024
map_dies 3026
report_differential_voltage 3028
report_irdrop_regions 3029
report_noise_margin 3032
extract_package 3034
report_esd_network 3036
report_esd_voltage 3038
report_joule_heat 3040
report_package 3042
report_rail 3045
report_resistance 3046
report_self_heat 3052
report_signal_resistance 3054
scale_what_if_capacitance 3056
scale_what_if_current 3059
scale_what_if_resistance 3061
set_advanced_package_options 3063
set_advanced_rail_options 3064
set_die_model 3065
set_dynamic_rail_simulation 3067
set_multi_die_analysis_mode 3069
set_net_group 3071
set_off_chip_package_trace 3072

set_package 3074
set_pg_nets 3077
set_power_data 3079
set_power_pads 3083
set_rail_analysis_config 3087
set_rail_analysis_domain 3137
set_self_heat_analysis_mode 3139
set_voltage_regulator_module 3141
view_package_results 3143
write_current_region 3144
write_die_model 3147
write_hier_view 3155
write_power_pads 3160
write_power_up_report 3164
32 3167
RC Extraction Commands and Attributes 3167
create_parasitic_coordinate_transform 3167
extract_rc 3171
extract_rc Category Attributes 3172
extract_rc_assume_metal_fill 3172
extract_rc_cap_filter_mode 3173
extract_rc_cerebrus_license_only 3173
extract_rc_compress_rcdb 3174
extract_rc_coupled 3174
extract_rc_coupling_cap_threshold 3174
extract_rc_def_via_cap 3174
extract_rc_effort_level 3175
extract_rc_engine 3175
extract_rc_extra_cmd_file 3175
extract_rc_hard_block_obs 3176
extract_rc_incremental 3176
extract_rc_layer_independent 3177
extract_rc_lef_tech_file_map 3177
extract_rc_local_cpu 3177
extract_rc_pvs_fill 3178
extract_rc_qrc_cmd_file 3178

extract_rc_qrc_cmd_type 3178
extract_rc_qrc_output_mode 3179
extract_rc_qrc_run_mode 3179
extract_rc_qrc_stream_map_file 3180
extract_rc_relative_cap_threshold 3180
extract_rc_shrink_factor 3181
extract_rc_signoff_stream_layer_map 3181
extract_rc_total_cap_threshold 3182
extract_rc_tquantus_model_file 3182
extract_rc_tsv_subckt_file 3182
extract_rc_turbo_reduce 3183
extract_rc_use_qrc_oa_interface 3183
extract_rc_use_shielding_in_detail_mode 3184
extract_rc_via_cap 3184
read_parasitics 3184
read_rc 3186
read_spef 3187
report_net_parasitics 3192
report_rcdb 3193
report_rc_factors 3194
report_unit_parasitics 3198
reset_parasitics 3200
set_extract_shrink_factor 3201
write_cap_table 3202
write_extraction_spec 3205
write_parasitics 3206
write_rcdb 3209

33 3211

Route Commands and Attributes 3211

add_tracks 3212
check_tquantus_model_file 3217
check_tracks 3218
colorize_cell_library 3220
create_channel_density_control 3221
create_tquantus_model_file 3222
delete_shields 3223

fix_drc_with_patch 3224
get_route_attributes 3225
report_congested_area 3226
report_route 3229
report_shields 3234
report_tracks 3236
report_wires 3237
route_add_shields 3240
route_design 3241
route_detail 3246
route_early_global 3247
route_early_global Category Attributes 3249
- route_early_global_effort_level 3249
- route_early_global_honor_partition_allow_feedthru 3249
- route_early_global_honor_partition_fence 3250
- route_early_global_honor_partition_pin 3250
- route_early_global_honor_partition_pin_guide 3250
- route_early_global_honor_power_domain 3251
- route_early_global_horizontal_supply_scale_factor 3251
- route_early_global_num_tracks_per_clock_wire 3251
- route_early_global_reverse_direction_regions 3251
- route_early_global_route_selected_net_only 3252
- route_early_global_secondary_pg 3252
- route_early_global_secondary_pg_max_fanout 3252
- route_early_global_stripe_layer_range 3252
- route_early_global_vertical_supply_scale_factor 3253

route_eco 3254
route_fix_signoff_drc 3257
route_global_detail 3258
route Category Attributes 3259
- extract_keep_fill_wires 3261
- route_add_antenna_inst_prefix 3261
- route_adjust_auto_via_weight 3261
- route_allow_inst_overlaps 3262
- route_allow_pin_as_feedthru 3262
- route_antenna_cell_name 3263
- route_antenna_diode_insertion 3263

route_concurrent_minimize_via_count_effort 3263
route_connect_to_bumps 3264
route_detail_add_passive_fill_only_on_layers 3264
route_detail_allow_passive_fill_only_in_layers 3264
route_detail_antenna_eco_list_file 3264
route_detail_auto_stop 3265
route_detail_check_mar_on_cell_pin 3265
route_detail_end_iteration 3265
route_detail_fix_antenna 3266
route_detail_fix_antenna_on_secondary_pg_nets 3266
route_detail_fix_antenna_with_gate_array_filler_mode 3266
route_detail_merge_abutting_cut 3266
route_detail_min_length_for_spread_wire 3267
route_detail_min_length_for_widen_wire 3267
route_detail_min_slack_for_opt_wire 3268
route_detail_no_taper_in_layers 3268
route_detail_no_taper_on_output_pin 3268
route_detail_on_grid_only 3269
route_detail_post_route_litho_repair 3269
route_detail_post_route_spread_wire 3270
route_detail_post_route_swap_via 3270
route_detail_post_route_via_pillar_effort 3271
route_detail_post_route_wire_widen 3271
route_detail_post_route_wire_widen_rule 3272
route_detail_postroute_via_priority 3272
route_detail_search_and_repair 3272
route_detail_signoff_effort 3272
route_detail_stub_routing_in_first_layer 3273
route_detail_use_multi_cut_via_effort 3273
route_diode_insertion_for_clock_nets 3274
route_eco_ignore_existing_route 3274
route_enable_route_rule_si_limit_length 3274
route_enforce_route_rule_on_special_net_wire 3275
route_extra_via_enclosure 3275
route_fix_clock_nets 3275
route_high_freq_constraint_groups 3276
route_high_freq_match_report_file 3276

route_high_freq_num_reserved_layers 3276
route_high_freq_remove_floating_shield 3276
route_high_freq_search_repair 3277
route_high_freq_shield_trim_length 3277
route_honor_exclusive_region 3277
route_honor_power_domain 3277
route_ignore_antenna_top_cell_pin 3278
route_ignore_follow_pin_shapes 3278
route_interposer_allow_diagonal_trunk 3278
route_interposer_interlayer_shielding_layers 3279
route_interposer_interlayer_shielding_nets 3279
route_interposer_interlayer_shielding_offsets 3279
route_interposer_interlayer_shielding_widths 3279
route_interposer_same_layer_shielding_net 3280
route_interposer_same_layer_shielding_width_spacing 3280
route_interposer_trunk_routing_layers 3280
route_interposer_trunk_routing_width_spacing 3281
route_number_fail_limit 3281
route_number_thread 3281
route_number_warning_limit 3282
route_process_node 3282
route_rc_extraction_corner 3283
route_relaxed_route_rule_spacing_to_power_ground_nets 3283
route_reserve_space_for_multi_cut 3283
route_reverse_direction 3284
route_route_clock_nets_first 3284
route_selected_net_only 3284
route_shield_crosstie_offset 3285
route_shield_report_skip_status 3285
route_shield_tap_cell_insertion 3285
route_shield_tap_cell_name 3285
route_skip_analog 3286
route_strictly_honor_1d_routing 3286
route_strict_honor_route_rule 3286
route_stripe_layer_range 3287
route_third_party_data 3287
route_tieoff_to_shapes 3287

route_trim_pull_back_distance_from_boundary 3288
route_trunk_with_cluster_target_size 3288
route_use_auto_via 3289
route_via_weight 3289
route_with_eco 3290
route_with_litho_driven 3290
route_with_si_driven 3290
route_with_timing_driven 3291
route_with_trim_metal 3291
route_with_via_in_pin 3292
route_with_via_only_for_block_cell_pin 3292
route_with_via_only_for_stdcell_pin 3292
route_trial_max_print_ignored_pad_nets 3293
set_max_route_layer 3294
set_route_attributes 3295
write_congested_area_report 3302
write_litho_hotspots 3303
write_nets_in_congested_area 3304
34 3306
Signal Integrity Commands and Attributes 3306
analyze_ir_paths 3307
check_noise 3310
define_bundled_bus 3313
fix_ac_limit_violations 3315
get_glitch_threshold 3319
report_bundled_bus 3320
report_double_clocking 3322
report_instance_cdb 3325
report_noise 3327
report_noise_lib_pin 3338
report_noise_propagation 3339
report_voltage_scaling 3341
report_voltage_swing 3343
reset_glitch_derate 3344
reset_noise_lib_pin 3345
set_annotated_glitch 3346

set_cdb_binding 3349
set_glitch_derate 3350
set_glitch_threshold 3352
set_model_priority 3355
set_mutex_condition 3356
set_noise_lib_pin 3357
set_outbound_report 3359
set_quiet_aggressor 3360
si Category Attributes 3362
- si_accumulated_small_aggressor_factor 3364
- si_accumulated_small_aggressor_mode 3364
- si_accumulated_small_aggressor_threshold 3364
- si_aggressor_alignment 3365
- si_clock_synchronicity 3365
- si_delay_clock_delta_threshold 3365
- si_delay_delta_threshold 3365
- si_delay_delta_annotation_mode 3365
- si_delay_enable_double_clocking_check 3366
- si_delay_enable_logical_correlation 3366
- si_delay_enable_report 3366
- si_delay_separate_on_data 3366
- si_enable_bus_attacker_correlation 3366
- si_enable_glitch_overshoot_undershoot 3367
- si_enable_two_stage_driver_weakening 3367
- si_enable_drv_with_delta_slew 3367
- si_enable_glitch_propagation 3368
- si_enable_glitch_propagation_spice_deck 3368
- si_enable_virtual_attacker_constituent_report 3368
- si_glitch_accuracy_level 3368
- si_glitch_analysis_type 3368
- si_glitch_constrained_input_threshold_failure_point 3369
- si_glitch_enable_dynamic_receiver_peak_limits 3369
- si_glitch_enable_report 3369
- si_glitch_input_threshold 3370
- si_glitch_input_voltage_high_threshold 3370
- si_glitch_input_voltage_low_threshold 3370
- si_glitch_receiver_clock_peak_limit 3370

si_glitch_receiver_latch_peak_limit 3371
si_glitch_receiver_peak_limit 3371
si_individual_aggressor_clock_threshold 3371
si_individual_aggressor_simulation_filter 3371
si_individual_aggressor_threshold 3371
si_max_virtual_attacker_constituents 3372
si_nonlinear_aggressor_slew 3372
si_num_iteration 3372
si_reselection 3372
si_reselection_delay_threshold 3373
si_reselection_hold_slack 3373
si_reselection_setup_slack 3373
si_secondary_attacker_decoupling_factor 3373
si_skip_noise_model_check 3373
si_skip_timing_window 3373
si_switch_probability 3374
si_pessimistic_mode 3374
si_unconstrained_net_use_infinite_timing_window 3374
si_use_infinite_timing_window 3375
update_glitch 3375
write_spice_deck 3377
35 3390
Structured Data Path Commands and Attributes 3390
add_sdp_rows_blockages 3390
check_sdps 3394
create_sdp 3398
delete_sdp_rows_blockages 3404
place_sdp Category Attributes 3407
place_sdp_clock_location 3407
place_sdp_disable_extend_core 3408
place_sdp_honor_dont_use 3408
place_sdp_honor_orient 3409
place_sdp_legalization 3409
place_sdp_legalization_effort 3410
place_sdp_max_move_distance 3410
place_sdp_num_column 3411

place_sdp_place_report 3411
place_sdp_pre_fixed_cells_blockage_direction 3412
place_sdps 3413
read_sdp_file 3416
set_sdp_origin 3418
update_sdp 3420
write_sdp_file 3429
36 3433
Integrated Hierarchical Database Commands 3433
check_module_model 3434
define_add_on_module_model 3437
import_module_model 3441
pack_module_model_libs 3443
read_module_model 3445
report_module_model 3449
set_module_model 3451
update_module_model 3457
write_module_model 3460
write_module_model_context 3463
37 3467
Timing Analysis Commands and Attributes 3467
Path Exception Priorities 3471
Examples of Path Exception Priorities 3472
Bidirectional Pin Defaults 3473
Defaults for Pin (or Port) Constraints 3473
Defaults for Port Constraints 3474
check_ldb_version 3474
check_timing 3476
check_timing_library 3494
check_timing_library_consistency 3496
create_analysis_view 3497
create_constraint_mode 3499
create_delay_corner 3501
create_library_set 3506
create_opcond 3510
create_rc_corner 3512

create_timing_condition 3518
delete_analysis_view 3520
get_capacitance_unit 3521
get_clock_network_objects 3522
get_constant 3523
get_equivalent_cells 3525
get_interactive_constraint_modes 3526
get_propagated_clock 3527
get_resistance_unit 3528
get_socv_reporting_nsigma_multiplier 3529
get_time_unit 3530
gui_show_timing_map 3531
merge_timing_reports 3531
read_path_descriptions 3533
read_sdf 3535
read_transition_file 3541
read_twf 3543
report_analysis_coverage 3545
report_analysis_summary 3555
report_analysis_views 3558
report_annotated_assertions 3560
report_annotated_check 3562
report_annotated_delay 3568
report_annotated_parasitics 3571
report_aocv_derate 3575
report_case_analysis 3579
report_cell_instance_timing 3584
report_clock_gating_check 3590
report_clock_propagation 3593
report_clock_timing 3596
report_clocks 3612
report_constraint 3622
report_cppr 3638
report_design 3642
report_fanin 3646
report_fanout 3649
report_inactive_arcs 3652

report_instance_library 3657
report_min_pulse_width 3660
report_mode 3666
report_nets 3668
report_path_based_aocv_derate 3672
report_path_exceptions 3675
report_path_groups 3680
report_ports 3682
report_socv_interconnect_variation 3688
report_socv_library 3689
report_statistical_timing_derate_factors 3691
report_timing 3694
report_timing_derate 3739
report_timing_summary 3744
report_wire_load 3747
reset_instance_library 3751
reset_path_adjust 3752
reset_path_adjust_group 3753
reset_sdf_assertions 3754
reset_timing_derate 3755
reset_wire_load_mode 3757
reset_wire_load_model 3758
reset_wire_load_selection_group 3760
set_analysis_view 3761
set_aocv_thresholds 3765
set_default_view 3766
set_instance_library 3768
set_interactive_constraint_modes 3769
set_io_thresholds 3771
set_path_adjust 3774
set_path_adjust_group 3776
set_socv_rc_variation_factor 3778
set_socv_reporting_nsigma_multiplier 3779
set_table_style 3781
set_timing_derate 3790
set_wire_load_mode 3798
set_wire_load_model 3801

set_wire_load_selection_group 3803
time_design 3805
time_design_signoff 3813
timing Category Attributes 3815
timing_all_registers_include_icg_cells 3824
timing_allow_input_delay_on_clock_source 3825
timing_analysis_aocv 3825
timing_analysis_async_checks 3825
timing_analysis_case_analysis 3826
timing_analysis_check_type 3826
timing_analysis_clock_gating 3826
timing_analysis_clock_net_marking_mode 3826
timing_analysis_clock_propagation_mode 3826
timing_analysis_clock_source_paths 3827
timing_analysis_cppr 3827
timing_analysis_engine 3828
timing_analysis_honor_active_logic_view 3828
timing_analysis_self_loops_paths_no_skew 3828
timing_analysis_precision_ps 3829
timing_analysis_socv 3829
timing_analysis_type 3829
timing_aocv_analysis_mode 3830
timing_aocv_chip_size 3830
timing_aocv_core_size 3831
timing_aocv_derate_mode 3831
timing_aocv_stage_count_update_on_timing_reset 3831
timing_apply_check_derate_to_external_output_delay 3832
timing_apply_default_primary_input_assertion 3832
timing_apply_exceptions_to_data_check_related_pin 3832
timing_apply_setup_hold_exceptions_to_data_check_related_pin 3833
timing_cap_unit 3834
timing_case_analysis_propagation 3834
timing_case_analysis_for_sequential_propagation 3834
timing_case_analysis_for_icg_propagation 3834
timing_check_timing_report_all_checks 3835
timing_check_timing_signal_level_high_to_low_threshold 3835
timing_check_timing_signal_level_low_to_high_threshold 3836

timing_clock_phase_propagation 3836
timing_clock_source_paths_unconstrained_mark_clock_used_as_data 3836
timing_clock_source_use_driving_cell 3837
timing_clock_uncertainty_from_to_precedence 3837
timing_collection_all_fanin_fanout_traversal_mode 3837
timing_collection_result_display_limit 3838
timing_collection_variable_assignment_compatibility 3838
timing_constraint_disable_min_max_input_delay_worst_casing 3838
timing_constraint_enable_detailed_report_invalid_begin_end_points 3838
timing_constraint_enable_drv_limit_override 3839
timing_constraint_enable_logging 3839
timing_constraint_enable_report_invalid_begin_end_points 3839
timing_constraint_enable_search_path 3840
timing_constraint_enable_separate_multicycle_data_checks 3840
timing_constraint_load_minimal_set_for_automated_eco 3840
timing_constraint_path_delay_exclude_check_delay_from_ignore_clock_latency 3840
timing_constraint_path_delay_exclude_io_delay_from_ignore_clock_latency 3841
timing_constraint_path_delay_exclude_unconstrained_endpoints 3841
timing_constraint_path_delay_include_clock_pin_endpoints 3841
timing_constraint_warn_for_timing_derate_exceeding_max_limit 3842
timing_continue_on_error 3842
timing_constraint_update_io_latency_averaging_mode 3842
timing_context_clock_mapping_percentage_tolerance 3843
timing_context_clock_mapping_check_waveform 3843
timing_cppr_enable_mismatch_transition_mode 3843
timing_cppr_opposite_edge_mean_scale_factor 3844
timing_cppr_opposite_edge_sigma_scale_factor 3844
timing_cppr_opposite_edge_sigma_scale_factor_net 3845
timing_cppr_opposite_edge_sigma_scale_factor_cell 3845
timing_cppr_propagate_thru_latches 3845
timing_cppr_remove_clock_to_data_pessimism 3845
timing_cppr_self_loop_mode 3846
timing_cppr_skip_clock_reconvergence 3846
timing_cppr_skip_clock_reconvergence_for_unmatched_clocks 3846
timing_cppr_threshold_ps 3846
timing_cppr_transition_sense 3847
timing_create_clock_default_propagated 3847

timing_default_opcond_per_lib 3848
timing_defer_mmmc_obj_updates 3848
timing_derate_aocv_dynamic_delays 3848
timing_derate_aocv_reference_point 3849
timing_derate_dynamic_compatibility 3849
timing_derate_ocv_reference_point 3849
timing_derate_negative_delay_backward_compatibility 3849
timing_derate_spatial_distance_unit 3850
timing_derate_incremental_adjust_additive_mode 3850
timing_disable_bus_contention_check 3850
timing_disable_clock_period_checks 3851
timing_disable_constant_propagation_for_sequential_cells 3851
timing_disable_drv_report_on_constant_nets 3851
timing_disable_floating_bus_check 3851
timing_disable_genclk_combinational_blocking 3852
timing_disable_inferred_clock_gating_checks 3852
timing_disable_inout_output_side_timing_checks 3852
timing_disable_internal_inout_cell_paths 3852
timing_disable_internal_inout_net_arcs 3853
timing_disable_lib_pulse_width_checks 3853
timing_disable_library_data_to_data_checks 3853
timing_disable_library_tieoffs 3853
timing_disable_netlist_constants 3854
timing_disable_nochange_checks 3854
timing_disable_non_sequential_checks 3854
timing_disable_output_as_clock_port 3855
timing_disable_pulse_width_same_edge_si_cppr_mode 3855
timing_disable_report_header_info 3855
timing_disable_retime_clock_path_slew_propagation 3856
timing_disable_sdf_retain_arc_merging 3856
timing_disable_skew_checks 3856
timing_disable_test_signal_arc 3856
timing_disable_timing_model_latch_inferencing 3857
timing_disable_tristate_disable_arcs 3857
timing_disable_user_data_to_data_checks 3858
timing_driving_cell_override_library 3858
timing_enable_aocv_slack_based 3858

timing_enable_case_analysis_conflict_warning 3858
timing_enable_clock_to_clock_clock_gating_check 3859
timing_enable_clock_phase_based_rise_fall_derating 3859
timing_enable_data_through_clock_gating 3859
timing_enable_derating_for_pulse_width_checks 3860
timing_enable_early_late_data_slews_for_setuphold_mode_checks 3860
timing_enable_genclk_divide_by_inherit_parent_duty_cycle 3860
timing_enable_genclk_source_path_register_limit 3861
timing_enable_generated_clock_edge_based_source_latency 3861
timing_enable_get_obj_escaped_name_backward_compatible 3861
timing_enable_get_objects_regexp_compatibility 3862
timing_enable_hierarchical_get_nets_support 3863
timing_enable_latch_thru_mode 3863
timing_enable_latency_through_clock_gating 3863
timing_enable_mmmc_loop_breaking 3864
timing_enable_multi_drive_net_reduction_with_assertions 3865
timing_enable_multi_frequency_latch_analysis 3865
timing_enable_multicycle_data_check_compatibility 3865
timing_enable_path_delay_to_unconstrained_endpoints_compatibility 3865
timing_enable_pessimistic_cppr_for_reconvergent_clock_paths 3866
timing_enable_power_ground_constants 3866
timing_enable_preset_clear_arcs 3867
timing_enable_pulse_latch 3867
timing_enable_si_cppr 3868
timing_enable_simultaneous_setup_hold_mode 3868
timing_enable_timing_window_pessimism_removal 3869
timing_enable_tristate_clock_gating 3869
timing_enable_uncertainty_for_clock_checks 3869
timing_enable_uncertainty_for_pulse_width_checks 3870
timing_extract_model_disable_cycle_adjustment 3870
timing_extract_model_aocv_mode 3870
timing_extract_model_case_analysis_in_library 3871
timing_extract_model_check_arcs_as_lvf 3871
timing_extract_model_consider_design_level_drv 3871
timing_extract_model_exhaustive_validation_dir 3872
timing_extract_model_exhaustive_validation_mode 3872
timing_extract_model_gating_as_nochange_arc 3872

timing_extract_model_ideal_clock_latency_arc 3873
timing_extract_model_include_applied_load_in_characterization_range 3873
timing_extract_model_include_applied_slew_in_characterization_range 3873
timing_extract_model_non_borrowing_latch_path_as_setup 3873
timing_extract_model_slew_propagation_mode 3873
timing_extract_model_write_clock_checks_as_arc 3874
timing_extract_model_write_clock_checks_as_scalar_tables 3874
timing_extract_model_write_lvf 3874
timing_extract_model_write_min_max_clock_tree_path 3874
timing_generated_clocks_inherit_ideal_latency 3875
timing_generated_clocks_allow_nested_assertions 3875
timing_generate_normalized_driver_waveform 3875
timing_get_of_objects_hier_compatibility 3875
timing_hier_obj_name_compatibility 3876
timing_ignore_lumped_rc_assertions 3876
timing_io_use_clock_network_latency 3877
timing_library_build_async_deassert_arc 3877
timing_library_convert_async_setuphold_to_recrem 3877
timing_library_generated_clock_use_group_name 3877
timing_library_hold_constraint_corner_sigma_multiplier 3878
timing_library_hold_sigma_multiplier 3878
timing_library_infer_async_pins_from_timing_arcs 3879
timing_library_infer_cap_range_from_ccs_receiver_model 3879
timing_library_infer_cap_range_from_ecsm_receiver_model 3879
timing_library_interpolate_drv_values 3879
timing_library_infer_socv_from_aocv 3880
timing_library_read_ccs_noise_data 3880
timing_library_read_without_ecsm 3880
timing_library_read_without_power 3880
timing_library_read_without_sensitivity 3880
timing_library_scale_aocv_to_socv_to_n_sigma 3881
timing_library_setup_constraint_corner_sigma_multiplier 3881
timing_library_setup_sigma_multiplier 3881
timing_library_term_voltage_from_lib_pin 3881
timing_library_use_two_piece_receiver_cap 3881
timing_library_zero_negative_timing_check_arcs 3882
timing_multi_frequency_clock_rounding_factor 3882

timing_normalized_driver_waveform_clip_linear_part 3882
timing_normalized_driver_waveform_weight_factor 3882
timing_nsigma_multiplier 3883
timing_null_collection_return_compatibility 3883
timing_path_based_enable_bounding_box_for_io_paths 3883
timing_path_based_exhaustive_pba_bounded_mode 3883
timing_path_based_enable_closest_common_pin_driver_for_bounding_box 3884
timing_path_based_enable_exhaustive_depth_bounded_by_gba 3884
timing_path_based_enable_report_launch_clock_path 3884
timing_path_based_enable_verbose_mode 3884
timing_path_based_exhaustive_enable_design_coverage 3885
timing_path_based_exhaustive_max_paths_limit 3885
timing_pba_exhaustive_path_nworst_limit 3886
timing_pll_clock_use_driving_cell 3886
timing_prefix_module_name_with_library_generated_clock 3887
timing_propagate_latch_data_uncertainty 3887
timing_property_arrival_clocks_consider_clock_source_data_phase 3888
timing_property_arrival_clocks_consider_data_phase 3888
timing_property_arrival_clocks_consider_latency_phase 3888
timing_property_arrival_window_enable_tcl_dict_format 3888
timing_property_return_null_collection_with_quiet 3889
timing_rail_swing_checks_high_voltage_threshold 3889
timing_rail_swing_checks_low_voltage_threshold 3889
timing_recompute_sdf_in_setuphold_mode 3889
timing_reduce_multi_drive_net_arcs 3890
timing_reduce_multi_drive_net_arcs_threshold 3890
timing_report_analysis_summary_csv_extended_new_format 3890
timing_report_check_timing_unconstrained_endpoints_due_to_constants 3891
timing_report_clock_pin_as_begin_point 3891
timing_report_constraint_enable_extended_drv_format 3891
timing_report_constraint_rise_fall_clock_period_check 3892
timing_report_disable_max_paths_per_group 3892
timing_report_drv_enable_clock_source_as_clock 3893
timing_report_drv_enable_frequency_per_view 3893
timing_report_drv_enable_slew_threshold_scaling 3893
timing_report_enable_cppr_point 3893
timing_report_enable_flag_field_symbols 3894

timing_report_enable_lead_trail_to_rise_fall_map 3894
timing_report_enable_markers 3894
timing_report_enable_max_capacitance_drv_for_constant_nets 3894
timing_report_enable_max_path_limit_warning 3894
timing_report_enable_report_clock_timing_across_clock_pin 3895
timing_report_enable_si_debug 3895
timing_report_enable_unique_pins_multiple_capture_clock_paths 3895
timing_report_enable_verbose_ssta_mode 3895
timing_report_fields 3896
timing_report_generated_clock_info 3896
timing_report_group_based_mode 3896
timing_report_max_transition_check_using_nsigma_slew 3898
timing_report_property_fastest_clock_consider_data_phase 3898
timing_report_pulse_width_matching_launch_capture_paths 3898
timing_report_redirect_message_types 3899
timing_report_retime_formatting_mode 3899
timing_report_skip_constraint_loop_check 3899
timing_report_socv_summary_mean_sigma 3899
timing_report_timing_header_detail_info 3900
timing_report_unconstrained_paths 3900
timing_report_unconstrained_path_early_late_header 3901
timing_report_use_receiver_model_capacitance 3901
timing_report_use_worst_parallel_cell_arc 3901
timing_resolve_driver_conflicts 3902
timing_scaling_for_negative_checks 3902
timing_scaling_for_negative_delays 3902
timing_sdf_adjust_negative_setuphold 3903
timing_sdf_enable_setuphold_scond_ccond 3903
timing_self_loop_paths_no_skew_max_depth 3904
timing_self_loop_paths_no_skew_max_slack 3904
timing_set_clock_source_to_output_as_data 3904
timing_socv_preserve_variation_with_annotations 3905
timing_socv_rc_variation_mode 3905
timing_socv_statistical_min_max_mode 3905
timing_socv_view_based_nsigma_multiplier_mode 3906
timing_spatial_derate_chip_size 3906
timing_spatial_derate_distance_mode 3906

timing_enable_spatial_derate_mode 3907
timing_suppress_escape_characters 3907
timing_suppress_ilm_constraint_mismatches 3907
timing_time_unit 3907
timing_use_clock_pin_attribute_for_clock_net_marking 3908
timing_use_incremental_si_transition 3908
timing_use_latch_early_launch_edge 3908
timing_use_latch_time_borrow 3908
timing_use_verilog_for_model_netlist 3909
timing_waveform_aware_pulse_width_checks_high_voltage_level 3909
timing_waveform_aware_pulse_width_checks_low_voltage_level 3909
timing_write_sdf_no_escape_backslash 3909
update_analysis_view 3910
update_constraint_mode 3912
update_delay_corner 3914
update_io_latency 3919
update_library_set 3920
update_opcond 3923
update_rc_corner 3924
update_timing_condition 3929
write_combinational_loop_break_constraints 3931
write_global_slack_report 3933
write_ldb 3938
write_path_descriptions 3940
write_sdc 3941
write_sdf 3944
write_timing_windows 3958
set_socv_constraint_config 3962
get_socv_rc_variation_factor 3965
create_property_alias 3966
get_clock_to_clock_association 3967
report_lib_arcs 3969
list_libraries 3975
38 3977
Timing Budgeting Commands and Attributes 3977
budget Category Attributes 3977

budget_abutted 3978
budget_accumulated 3978
budget_boundary_condition_template_path 3979
budget_buffer_delay_adjustment 3979
budget_buffer_delay_lib_cell 3979
budget_buffer_delay_selection_effort 3980
budget_constant_model 3980
budget_distribute_mmmc 3980
budget_drive_cell 3980
budget_fix_top_level_paths 3980
budget_handle_complex_sdc 3980
budget_honor_report_timing_format 3981
budget_ignore_dont_touch 3981
budget_include_latency 3981
budget_include_wire_loads_in_lib 3981
budget_input_load 3981
budget_input_transition 3981
budget_justify 3982
budget_keep_pin_list_for_block_ports 3982
budget_latency_on_clocks 3982
budget_local_latency 3982
budget_local_uncertainty 3983
budget_make_negative_input_delay_zero 3983
budget_master_clone 3983
budget_no_false_paths_for_unconstrained_ports 3983
budget_no_hold_view 3983
budget_no_setup_view 3983
budget_override_net_cap 3983
budget_pin_load 3984
budget_report_dir 3984
budget_report_negative_slack_on_ports 3984
budget_report_or_update_budget 3984
budget_snap_feedthru_budget_to 3984
budget_snap_input_budget_to 3984
budget_snap_negative_only 3985
budget_snap_output_budget_to 3985
budget_stage_based_fanout_driving_factor 3985

budget_stage_based_partition_multiplier 3986
budget_stage_based_weight 3986
budget_top_level 3986
budget_top_level_delay_per_length 3986
budget_top_level_min_delay_per_net 3987
budget_use_boundary_condition 3987
budget_use_real_cell_for_timing_model 3987
budget_virtual_opt_engine 3987
budget_write_constraints_for_clock_output_ports 3988
budget_write_false_path_for_hold 3988
budget_write_latency_per_clock 3988
check_partition_sdc 3988
create_timing_budget 3991
report_timing_budget 3994
report_timing_budget_exception 3996
reset_modified_budget 3997
set_cycle_budget_ratio 3998
set_fixed_budget 4002
set_partition_user_constraints_file 4004
update_timing_budget 4005
write_timing_budgets 4006
39 4008
Timing Constraint Commands 4008
create_clock 4011
create_generated_clock 4014
current_design 4018
current_instance 4018
get_current_constraint_mode 4020
group_path 4021
reset_annotated_check 4023
reset_annotated_delay 4025
reset_annotated_transition 4026
reset_aocv_stage_weight 4027
reset_case_analysis 4028
reset_clock 4029
reset_clock_gating_check 4031

reset_clock_groups 4032
reset_clock_latency 4034
reset_clock_sense 4037
reset_clock_transition 4038
reset_clock_tree_latency 4041
reset_clock_uncertainty 4042
reset_data_check 4045
reset_disable_clock_gating_check 4047
reset_disable_timing 4048
reset_drive 4051
reset_driving_cell 4052
reset_generated_clock 4053
reset_ideal_latency 4055
reset_ideal_network 4056
reset_ideal_transition 4057
reset_input_delay 4058
reset_load 4060
reset_max_capacitance 4061
reset_max_fanout 4062
reset_max_time_borrow 4063
reset_max_transition 4064
reset_min_capacitance 4065
reset_min_fanout 4066
reset_min_pulse_width 4067
reset_min_transition 4068
reset_mode 4069
reset_output_delay 4071
reset_path_exception 4075
reset_path_group 4080
reset_pll_timing 4081
reset_propagated_clock 4081
reset_property 4084
reset_pulse_clock_max_transition 4085
reset_pulse_clock_max_width 4086
reset_pulse_clock_min_transition 4087
reset_pulse_clock_min_width 4088
reset_resistance 4089

set_active_clocks 4090
set_annotated_check 4091
set_annotated_delay 4094
set_annotated_transition 4097
set_aocv_interface_path_offset 4098
set_aocv_stage_weight 4100
set_case_analysis 4101
set_clock_exclusivity 4102
set_clock_gating_check 4103
set_clock_groups 4106
set_clock_latency 4109
set_clock_sense 4112
set_clock_transition 4115
set_clock_uncertainty 4117
set_data_check 4120
set_disable_clock_gating_check 4122
set_disable_timing 4123
set_dont_touch 4125
set_dont_touch_network 4126
set_dont_use 4127
set_drive 4128
set_driving_cell 4130
set_false_path 4132
set_fanout_load 4136
set_ideal_latency 4137
set_ideal_network 4139
set_ideal_transition 4140
set_input_delay 4141
set_input_transition 4144
set_load 4146
set_logic_one 4149
set_logic_zero 4150
set_max_capacitance 4151
set_max_delay 4154
set_max_fanout 4159
set_max_time_borrow 4161
set_max_transition 4162

set_min_capacitance 4165
set_min_delay 4168
set_min_fanout 4175
set_min_pulse_width 4177
set_min_transition 4179
set_mode 4182
set_multicycle_path 4184
set_normalized_driver_waveform_lib 4190
set_output_delay 4191
set_pll_timing 4194
set_propagated_clock 4198
set_property 4200
set_pulse_clock_max_transition 4201
set_pulse_clock_max_width 4202
set_pulse_clock_min_transition 4203
set_pulse_clock_min_width 4204
set_resistance 4206
write_annotated_transition 4208
set_sense 4209
reset_sense 4211
40 4213
Timing Debug Commands 4213
analyze_paths_by_bottleneck 4214
analyze_paths_by_clock_domain 4217
analyze_paths_by_critical_false_path 4219
analyze_paths_by_default_path_group 4221
analyze_paths_by_drv 4223
analyze_paths_by_hier_port 4225
analyze_paths_by_hierarchy 4228
analyze_paths_by_view 4234
create_histogram_png 4236
create_path_category 4238
delete_path_category 4251
gui_gtd_highlight_timing_report 4252
gui_open_gtd 4254
read_path_categories 4255

read_timing_debug_report 4256
write_category_summary 4259
write_histogram_view 4262
write_path_categories 4263
write_path_list_summary 4264
write_text_timing_report 4265
41 4268
Timing Modeling Commands 4268
check_timing_model 4268
merge_model_timing 4276
report_timing_model 4278
write_timing_model 4284
42 4293
Timing Optimization Commands and Attributes 4293
add_repeater_by_rule 4295
add_tieoffs 4302
add_tieoffs Category Attributes 4306
add_tieoffs_cells 4306
add_tieoffs_create_hports 4307
add_tieoffs_honor_dont_touch 4307
add_tieoffs_honor_dont_use 4307
add_tieoffs_max_distance 4308
add_tieoffs_max_fanout 4308
add_tieoffs_module_prevention 4308
add_tieoffs_prefix 4309
add_tieoffs_report_hports 4309
check_footprint 4309
create_basic_path_groups 4311
delete_buffer_trees 4315
delete_tieoffs 4318
eco_clone 4320
eco_clone_flop 4326
eco_merge_combinational_cell 4329
eco_split_combinational_cell 4332
eco_split_flop 4335
eco_split_complex_flop 4340

get_scheduling_file 4343
opt_area 4344
opt_clock_skew 4346
opt_design 4351
opt_power 4367
opt_signoff 4371
opt_signoff Category Attributes 4375
- opt_signoff_add_inst 4378
- opt_signoff_add_load 4378
- opt_signoff_allow_skewing 4379
- opt_signoff_along_route_buffering 4380
- opt_signoff_buffer_cell_list 4380
- opt_signoff_check_drv_from_hold_views 4381
- opt_signoff_check_type 4381
- opt_signoff_clock_cell_list 4382
- opt_signoff_clock_max_level 4382
- opt_signoff_delete_inst 4383
- opt_signoff_disable_geometry_checks 4383
- opt_signoff_drv_margin 4384
- opt_signoff_eco_file_prefix 4384
- opt_signoff_em_type 4385
- opt_signoff_fix_cell_em 4385
- opt_signoff_fix_clock_drv 4386
- opt_signoff_fix_data_drv 4386
- opt_signoff_fix_glitch 4386
- opt_signoff_fix_hold_allow_setup_optimization 4387
- opt_signoff_fix_hold_allow_setup_tns_degrade 4387
- opt_signoff_fix_hold_with_margin 4387
- opt_signoff_fix_ir_drop 4388
- opt_signoff_fix_max_cap 4388
- opt_signoff_fix_osus_glitch 4389
- opt_signoff_fix_max_transition 4389
- opt_signoff_fix_slack_in_range_lower_limit 4389
- opt_signoff_fix_slack_in_range_upper_limit 4390
- opt_signoff_fix_si_slew 4390
- opt_signoff_fix_xtalk 4391
- opt_signoff_hold_target_slack 4391

opt_signoff_hold_xtalk_delta_threshold 4392
opt_signoff_hold_xtalk_slack_threshold 4392
opt_signoff_ignore_drv_checks 4392
opt_signoff_keep_tmp_files 4393
opt_signoff_legal_only 4393
opt_signoff_load_cell_list 4394
opt_signoff_max_cap_margin 4394
opt_signoff_max_transition_margin 4395
opt_signoff_max_opt_runtime 4395
opt_signoff_max_paths 4396
opt_signoff_max_runtime 4396
opt_signoff_max_slack 4397
opt_signoff_num_report_paths 4397
opt_signoff_nworst 4398
opt_signoff_optimize_core_only 4398
opt_signoff_optimize_replicated_modules 4399
opt_signoff_optimize_sequential_cells 4399
opt_signoff_partition_list_file 4400
opt_signoff_pba_eco_route 4400
opt_signoff_pba_effort 4401
opt_signoff_post_mask 4401
opt_signoff_post_mask_decap_cells 4402
opt_signoff_post_mask_filler_cells 4402
opt_signoff_post_mask_gate_array_filler_cells 4403
opt_signoff_post_mask_logic_cells 4403
opt_signoff_post_sta_tcl 4404
opt_signoff_power_aware 4404
opt_signoff_power_opt_focus 4405
opt_signoff_pre_sta_tcl 4406
opt_signoff_prefix 4406
opt_signoff_preserve_filler 4407
opt_signoff_read_eco_opt_db 4407
opt_signoff_read_irdrop_db 4408
opt_signoff_resize_inst 4408
opt_signoff_retime 4409
opt_signoff_retime_mode 4409
opt_signoff_retime_depth 4409

opt_signoff_retime_depth_early 4410
opt_signoff_routing_congestion_aware 4410
opt_signoff_select_hold_endpoints 4411
opt_signoff_select_drv_net_file 4411
opt_signoff_select_drv_pin_file 4411
opt_signoff_select_setup_endpoints 4412
opt_signoff_setup_recovery 4412
opt_signoff_setup_target_slack 4413
opt_signoff_setup_xtalk_delta_threshold 4413
opt_signoff_setup_xtalk_slack_threshold 4414
opt_signoff_skip_drv_net_file 4414
opt_signoff_set_hold_endpoints_margin 4415
opt_signoff_set_setup_endpoints_margin 4415
opt_signoff_swap_inst 4415
opt_signoff_verbose 4416
opt_signoff_write_eco_opt_db 4416
opt_virtual 4417
opt Category Attributes 4418
opt_activity_refresh_args 4422
opt_add_always_on_feed_through_buffers 4423
opt_add_insts 4423
opt_add_ports 4424
opt_add_repeater_report_failure_reason 4424
opt_all_end_points 4425
opt_allow_only_cell_swapping 4426
opt_allow_multi_bit_on_flop_with_sdc 4426
opt_area_recovery 4427
opt_area_recovery_setup_target_slack 4428
opt_concatenate_default_and_user_prefixes 4428
opt_consider_routing_congestion 4429
opt_constant_inputs 4430
opt_constant_nets 4430
opt_delete_insts 4430
opt_detail_drv_failure_reason 4431
opt_detail_drv_failure_reason_max_num_nets 4431
opt_down_size_insts 4432
opt_drv_margin 4432

opt_enable_data_to_data_checks 4433
opt_enable_podv2_clock_opt_flow 4433
opt_enable_restructure 4433
opt_drv 4434
opt_drv_with_miller_cap 4434
opt_fix_fanout_load 4434
opt_hold_allow_overlap 4435
opt_hold_allow_resize 4435
opt_hold_allow_setup_tns_degradation 4436
opt_hold_ignore_path_groups 4436
opt_hold_cells 4437
opt_hold_on_excluded_clock_nets 4437
opt_hold_slack_threshold 4438
opt_hier_add_antenna_cell 4438
opt_hier_opt_stage 4438
opt_high_effort_cells 4439
opt_hold_target_slack 4439
opt_honor_density_screen 4440
opt_honor_fences 4440
opt_leakage_to_dynamic_ratio 4441
opt_max_density 4441
opt_max_length 4442
opt_move_insts 4442
opt_multi_bit_combinational_mode 4443
opt_multi_bit_combinational_opt 4443
opt_multi_bit_combinational_merge_timing_effort 4444
opt_multi_bit_combinational_split_timing_effort 4444
opt_multi_bit_flop_merge_bank_label_inference 4445
opt_multi_bit_flop_merge_timing_effort 4445
opt_multi_bit_flop_opt 4446
opt_multi_bit_flop_name_prefix 4446
opt_multi_bit_flop_name_separator 4447
opt_multi_bit_flop_name_suffix 4448
opt_multi_bit_flop_reorder_bits 4448
opt_multi_bit_flop_split_report_failure_reason 4449
opt_multi_bit_flop_split_timing_effort 4449
opt_multi_bit_unused_bits 4450

opt_multi_bit_unused_bit_count 4451
opt_new_inst_prefix 4451
opt_new_net_prefix 4451
opt_one_pass_lec 4452
opt_podv2_flow_effort 4452
opt_pin_swapping 4453
opt_post_route_allow_overlap 4453
opt_post_route_area_reclaim 4453
opt_post_route_art_flow 4454
opt_post_route_check_antenna_rules 4455
opt_post_route_drv_recovery 4455
opt_post_route_fix_clock_drv 4456
opt_post_route_fix_glitch 4456
opt_post_route_fix_si_transitions 4457
opt_post_route_hold_recovery 4457
opt_post_route_report_si_transitions 4458
opt_post_route_setup_recovery 4458
opt_power_effort 4459
opt_pre_route_ndr_aware 4461
opt_preserve_all_sequential 4461
opt_preserve_hpin_function 4461
opt_remove_redundant_insts 4462
opt_resize_flip_flops 4463
opt_resize_level_shifter_and_iso_insts 4463
opt_resize_power_switch_insts 4463
opt_flop_pins_report 4464
opt_flops_report 4464
opt_sequential_genus_restructure_report_failure_reason 4465
opt_setup_target_slack 4465
opt_target_based_opt_file 4465
opt_target_based_opt_hold_file 4466
opt_target_based_opt_file_only 4467
opt_tied_inputs 4467
opt_time_design_compress_reports 4468
opt_time_design_expanded_view 4468
opt_time_design_num_paths 4468
opt_time_design_report_net 4469

opt_time_design_vertical_timing_summary 4469
opt_unfix_clock_insts 4470
opt_skew 4470
opt_skew_apply_delay_limits_to_full_flow 4471
opt_skew_ccopt 4471
opt_skew_post_route 4472
opt_skew_pre_cts 4472
opt_skew_delay_pre_cts 4473
opt_skew_macro_only 4473
opt_skew_max_allowed_delay 4474
opt_skew_min_allowed_delay 4474
opt_skew_no_boundary 4475
opt_verbose 4475
opt Category Object Attributes 4478
read_footprint 4480
report_always_on_buffer 4482
report_critical_nets 4484
report_critical_insts 4486
report_critical_pins 4491
report_footprints 4493
report_ignored_nets 4496
report_lef_equivalent_cells 4498
report_length_violations 4500
report_multibit 4502
report_multi_bit_ffs 4508
report_path_group_options 4515
report_route_type_constraints 4517
report_threshold_instance_count 4525
reset_path_group_options 4530
route_opt_design 4532
set_buffer_footprint 4537
set_delay_footprint 4539
set_inverter_footprint 4541
set_max_cap_per_freq 4543
set_max_cap_per_freq_transition 4547
set_max_transition_per_freq 4551
set_path_group_options 4555

set_scheduling_file 4566
write_multi_bit_flop_mapping_file 4567
write_target_based_opt_file 4572
43 4577
TSV Design Commands 4577
add_obj_on_bump 4578
add_tsv 4582
assign_tsv 4590
check_stacked_die 4593
create_stress_rule 4594
delete_tsv 4595
read_bump_locations 4596
read_codesign_die_abstract 4600
read_die_abstract 4601
read_tsv_config 4602
unassign_tsv 4604
write_bump_locations 4606
write_codesign_die_abstract 4609
write_die_abstract 4615
write_micro_bump_mapping_file 4616
write_spef_between_tsv_dies 4618
write_design_stack_config 4619
read_design_stack_config 4620
44 4623
Unix Commands 4623
flowtool 4623
innovus 4626
lef2oa 4634
nagelfar 4637
oa2lef 4639
45 4641
Unified Safety Format Commands 4641
USF File Commands 4641
apply_safety_mechanism 4642
create_esubpart 4644

create_failure_mode 4647
create_safety_mechanism 4650
set_safety_mechanism_rules_dcls 4653
set_safety_mechanism_rules_tmr 4657
USF Support Commands 4660
check_safety_mechanism 4660
read_safety_tmr 4663
read_usf 4666
reset_usf 4667
usf_version 4668
write_usf 4669
46 4671
Wire Edit Commands and Attributes 4671
convert_special_routes_for_eco 4672
create_fillet 4675
delete_fillet 4676
delete_routes 4677
delete_secondary_pg_net 4685
deselect_routes 4688
deselect_secondary_pg_net 4693
edit_add_polygon_point 4695
edit_add_route_point 4697
edit_add_trim_metal 4699
edit_adjust_routes 4702
edit_create_via 4709
edit_cut_route 4710
edit_duplicate_routes 4716
edit_end_polygon_point 4718
edit_end_route_point 4720
edit_merge_routes 4722
edit_move_routes 4724
edit_resize_routes 4729
edit_rotate 4737
edit_split_routes 4743
edit_split_wide_routes 4745
edit_trim_routes 4746

edit_update_route_layer 4751
edit_update_route_mask 4753
edit_update_route_net 4757
edit_update_route_rule 4759
edit_update_route_status 4761
edit_update_via 4764
edit_wire_via Category Attributes 4767
edit_wire_via_allow_geometry_drc 4767
edit_wire_via_auto_snap 4768
edit_wire_via_auto_update 4769
edit_wire_via_cell_name 4769
edit_wire_via_create_by 4770
edit_wire_via_cut_layer 4771
edit_wire_via_snap_honor_color 4772
edit_wire_via_type 4772
edit_wire Category Attributes 4774
edit_wire_align 4777
edit_wire_allow_45_degree 4778
edit_wire_arrow_incremental 4779
edit_wire_assign_multi_pattern_color 4780
edit_wire_auto_distribute_bus 4781
edit_wire_auto_split_bus 4782
edit_wire_bus_honor_start_parameters 4783
edit_wire_bus_honor_width_setting 4784
edit_wire_bus_total_width_horizontal 4785
edit_wire_bus_total_width_vertical 4786
edit_wire_change_order_at_turn 4787
edit_wire_check_design_boundary 4788
edit_wire_close_polygons 4789
edit_wire_color_align_with_track 4790
edit_wire_connect_pin 4791
edit_wire_connect_with_specified_layer 4792
edit_wire_create_crossover_vias 4793
edit_wire_create_is_edit_flag 4794
edit_wire_create_via_on_pin 4795
edit_wire_cut_class 4796
edit_wire_cut_wire_overlap 4797

edit_wire_debug_file 4798
edit_wire_delete_tsv 4798
edit_wire_delete_pin_with_wire 4799
edit_wire_delete_wire_via_through_layers 4800
edit_wire_display_wire_length_with_cursor 4800
edit_wire_draw_shield 4801
edit_wire_draw_with_group_centerline 4802
edit_wire_drawing_wire 4803
edit_wire_drc_aware_cross_metal 4804
edit_wire_drc_on 4805
edit_wire_drc_use_non_default_spacing 4805
edit_wire_extend_wires 4806
edit_wire_final_check_with_verify 4807
edit_wire_ignore_drc 4808
edit_wire_jog_connect_layer 4809
edit_wire_keep_floating_via 4810
edit_wire_keep_status 4811
edit_wire_keep_via 4813
edit_wire_lateral_movement_range 4814
edit_wire_layer 4818
edit_wire_layer_horizontal 4819
edit_wire_layer_max 4820
edit_wire_layer_min 4821
edit_wire_layer_vertical 4822
edit_wire_look_down_layers 4823
edit_wire_look_up_layers 4824
edit_wire_max_pointer_number 4825
edit_wire_mirror_bus_route 4826
edit_wire_nets 4827
edit_wire_no_merge_special_wire 4828
edit_wire_only_show_edit_layer 4829
edit_wire_orthogonal_connection_only 4830
edit_wire_outer_shield_spacing 4831
edit_wire_outer_shield_width 4832
edit_wire_override 4833
edit_wire_partial_overlap_threshold 4834
edit_wire_pull_back_distance 4836

edit_wire_reshape 4838
edit_wire_return_obj_pointer 4839
edit_wire_rule 4841
edit_wire_search_route_rule_vias_only 4842
edit_wire_shape 4843
edit_wire_shield 4844
edit_wire_shield_look_down_layers 4845
edit_wire_shield_look_up_layers 4846
edit_wire_shield_shape 4847
edit_wire_shielding_nets 4848
edit_wire_show_drc_info_for_edit_shape 4849
edit_wire_sibling_look_down_layers 4851
edit_wire_sibling_look_up_layers 4852
edit_wire_snap 4853
edit_wire_snap_align_to 4854
edit_wire_snap_bus_to_pin 4855
edit_wire_snap_end_to 4858
edit_wire_snap_objects_to_track 4859
edit_wire_snap_to 4860
edit_wire_snap_to_track_honor_color 4861
edit_wire_snap_trim_metal_to_trim_grid 4862
edit_wire_spacing 4863
edit_wire_spacing_horizontal 4864
edit_wire_spacing_vertical 4865
edit_wire_status 4866
edit_wire_stop_at_drc 4867
edit_wire_stretch_end 4868
edit_wire_stretch_with_intersection 4869
edit_wire_sub_class 4870
edit_wire_turn_at 4871
edit_wire_type 4873
edit_wire_unrestricted_regular_wire_width 4874
edit_wire_update_shield_net 4875
edit_wire_use_fix_via 4876
edit_wire_use_interleaving_wire_group 4877
edit_wire_use_wire_group 4878
edit_wire_use_wire_group_bits 4879

edit_wire_use_wire_group_reinforcement 4880
edit_wire_use_wire_group_reinforcement_group_via 4881
edit_wire_use_wire_group_reinforcement_spacing 4882
edit_wire_use_wire_group_reinforcement_width 4883
edit_wire_verbose 4884
edit_wire_via_columns 4885
edit_wire_via_exclude_spec 4886
edit_wire_via_override_spec 4886
edit_wire_via_rows 4888
edit_wire_via_scale_height 4889
edit_wire_via_scale_width 4890
edit_wire_via_snap_to_intersection 4891
edit_wire_width 4892
edit_wire_width_horizontal 4893
edit_wire_width_vertical 4894
edit_wire_wire_override_spec 4895
end_parallel_edit 4897
read_parallel_edit_files 4899
report_legal_wire_width_for_bump 4902
select_routes 4903
select_secondary_pg_net 4912
start_parallel_edit 4915
47 4918
Metric Commands and Attributes 4918
Common Metric Names 4919
Clock Metrics 4919
Design Metrics 4922
Flow Metrics 4937
Frequency Metrics 4939
Power Metrics 4940
Rail Metrics 4947
Route Metrics 4950
Timing Metrics 4953
create_metric_page 4962
create_metric_table 4964
create_metric_table_heading 4967

create_snapshot 4969
define_metric 4971
delete_metric 4978
delete_metric_page 4979
delete_metric_run 4980
delete_metric_table 4981
delete_metric_table_heading 4982
get_metric 4983
get_metric_alias 4985
get_metric_config 4986
get_metric_definition 4988
get_metric_header 4989
metric Category Attributes 4990
- metric_advanced_url_endpoint 4991
- metric_capture_depth 4991
- metric_capture_design_image 4991
- metric_capture_design_image_blockages 4991
- metric_capture_design_image_power_intent 4991
- metric_capture_design_image_route_drc 4991
- metric_capture_max_drc_markers 4992
- metric_capture_min_count 4992
- metric_capture_overwrite 4992
- metric_capture_pba_tns_histogram 4992
- metric_capture_timing_analysis_mode 4992
- metric_capture_per_view 4993
- metric_capture_reg2reg_metrics 4993
- metric_capture_timing_paths 4993
- metric_capture_timing_path_groups 4993
- metric_capture_tns_histogram 4993
- metric_capture_tns_histogram_buckets 4994
- metric_capture_tns_histogram_paths 4994
- metric_capture_tns_histogram_max_slack 4994
- metric_capture_vth_per_power_domain 4994
- metric_category_default 4994
- metric_current_run_id 4995
- metric_enable 4995
- metric_summary_metrics 4995

pop_snapshot_stack 4995
push_snapshot_stack 4997
read_metric 4999
read_metric_config 5001
read_metric_html_package 5002
report_metric 5003
report_qor 5005
reset_metric_config 5006
reset_metric_run_id 5007
run_metric_category 5008
set_metric 5009
set_metric_alias 5010
set_metric_header 5012
update_metric 5013
update_metric_definition 5015
write_metric 5020
write_metric_config 5022
rename_snapshot 5023
delete_metric_header 5024
Appendix A: Database Object Information 5026
Index 5027
A-Z 5027

About This Manual

The Cadence® Innovus™ Implementation System family of products provides an integrated solution for an RTL-to-GDSII design flow. This manual describes the syntax for the Innovus Stylus Common UI text commands. The chapters are presented alphabetically by general command functionality. Within each chapter, the text commands appear alphabetically.

This document applies to the Innovus Stylus Common UI text commands, including syntax and examples. See the *Innovus Text Command Reference* document for Innovus text commands.

Audience

This manual is written for experienced designers of digital integrated circuits. Such designers must be familiar with design planning, placement and routing, block implementation, chip assembly, and design verification. Designers must also have a solid understanding of UNIX and Tcl/Tk programming.

Conventions Used in This Manual

This section describes the typographic and syntax conventions used in this manual.

`text`	Indicates text that you must type exactly as shown. For example: `set_message -severity info`
`text`	Indicates information for which you must substitute a name or value. In the following example, you must substitute the name of a specific file for *`file_name`*: `compare_release -out_file` *`file_name`*
text	Indicates the following: • Text found in the graphical user interface (GUI), including form names, button labels, and field names • Terms that are new to the manual, are the subject of discussion, or need special emphasis • Titles of manuals

`[ ]`	Indicates optional arguments. In the following example, you can specify none, one, or both of the bracketed arguments: `command [-arg1] [arg2 value]`
`[ | ]`	Indicates an optional choice from a mutually exclusive list. In the following example, you can specify any of the arguments or none of the arguments, but you cannot specify more than one: `command [arg1 | arg2 | arg3 | arg4]`
`{ | }`	Indicates a required choice from a mutually exclusive list. In the following example, you must specify one, and only one, of the arguments: `command {arg1 | arg2 | arg3}`
`{[ ] [ ]}`	Indicates a required choice of one or more items in a list. In the following example, you must choose one argument from the list, but you can choose more than one: `command {[arg1] [arg2] [arg3]}`
`{ }`	Indicates curly braces that must be entered with the command syntax. In the following example, you must type the curly braces: `command arg1 {x y}`
. . .	Indicates that you can repeat the previous argument.
. . .	Indicates an omission in an example of computer output or input.
Command - Subcommand	Indicates a command sequence, which shows the order in which you choose commands and subcommands from the GUI menu. In the following example, you choose *Power* from the menu, then *Power Planning* from the submenu, and then *Add Ring* from the displayed list: *Power - Power Planning - Add Ring* This sequence opens the *Add Rings* form.

Related Documents

For more information about the Innovus family of products, see the following documents. You can access these and other Cadence documents with the Cadence Help documentation system.

Innovus Product Documentation

- What's New in Innovus
  Provides information about new and changed features in this release of the Innovus family of products.
- Innovus User Guide
  Describes how to install and configure the Innovus software, and provides strategies for implementing digital integrated circuits.
- Innovus Text Command Reference
  Describes the Innovus text commands, including syntax and examples.
- Innovus Menu Reference
  Provides information specific to the forms and commands available from the Innovus graphical user interface.
- Innovus Database Access Command Reference
  Lists all the Innovus database access commands and provides a brief description of syntax and usage.
- Innovus Foundation Flows User Guide
  Describes how to use the scripts that represent the recommended implementation flows for digital timing closure with the Innovus software.
- Mixed Signal Interoperability Guide
  Describes the digital mixed-signal flow.
- README file
  Contains installation, compatibility, and other prerequisite information, including a list of Cadence Change Requests (CCRs) that were resolved in this release. You can read this file online at downloads.cadence.com.

Innovus Stylus Common UI Documentation

- Innovus Stylus Common UI Migration Guide
  Provides information on migrating from legacy to Innovus Stylus Common UI version of the Innovus software.
- What's New in Innovus Stylus Common UI
  Provides information about new and changed features in this release of the Innovus family of products.
- Innovus Stylus Common UI User Guide
  Describes how to install and configure the Innovus Stylus Common UI software, and provides strategies for implementing digital integrated circuits.
- Innovus Stylus Common UI Menu Reference
  Provides information specific to the forms and commands available from the Innovus Stylus Common UI graphical user interface.
- Stylus Common UI Database Object Information
  Provides information about Stylus Common UI database objects.
- Innovus Stylus Common UI Mixed Signal (MS) Interoperability Guide
  Describes the digital mixed-signal flow using Innovus Stylus Common UI.

For a complete list of documents provided with this release, see the Cadence Help online documentation system.

1

Advanced Timing Tcl Scripting Commands

- add_to_collection
- all_clocks
- all_connected
- all_fanin
- all_fanout
- all_inputs
- all_instances
- all_outputs
- all_registers
- append_to_collection
- compare_collections
- copy_collection
- define_property
- filter_collection
- foreach_in_collection
- get_activity
- get_arcs
- get_cells
- get_clocks
- get_designs
- get_generated_clocks
- get_lib_arcs
- get_lib_cells
- get_lib_pg_pins
- get_lib_pins
- get_libs
- get_nets

- get_object_name
- get_path_groups
- get_pg_nets
- get_pg_pins
- get_pins
- get_ports
- get_power
- get_property
- index_collection
- query_objects
- range_collection
- remove_from_collection
- report_object_property_names
- report_property
- sizeof_collection
- sort_collection

Overview

The following lists the Advanced Timing Tcl Scripting commands. For each category of similar commands, we explain what they are used for followed by some simple examples.

all_* Commands

The `all_*` commands return a collection of all the items of a specific type. Several of the commands provide options to filter the collection based on certain properties:

- `all_clocks`
- `all_connected`
- `all_fanin`
- `all_fanout`
- `all_inputs`
- `all_instances`
- `all_outputs`
- `all_registers`

Examples

- To set all the clocks to propagated mode, use the following command:
  ```
  set_propagated_clock [all_clocks]
  ```
- To report the worst timing path that begins at a register and ends at an output port, use the following command:
  ```
  report_timing -from [all_registers] -to [all_outputs]
  ```
- To store a collection of pins that fanout from a specific port, use the following command:
  ```
  set clk1Fanout [all_fanout -from clk1 -pin_levels 1]
  ```

get_* Commands

The `get_*` commands return a collection of items filtered by name pattern or object type. Additionally, you can use the `-filter` option for most of the `get_*` commands to filter the collection by property value. See the get_property command for a list of all the properties.

The `filter_expression` for the `-filter` option can be created by combining several object properties using the following relational and logical operators: `>`, `<`, `==`, `!=`, `<=`, `>=`, `&&`, `||`, `=~`, `!~`, `AND`, and `OR`. You also can use parentheses to create expressions. The `get_*` commands include the following:

- `get_arcs`
- `get_cells`
- `get_clocks`
- `get_designs`
- `get_generated_clocks`
- `get_lib_arcs`
- `get_lib_cells`
- `get_lib_pins`
- `get_libs`
- `get_nets`
- `get_object_name`
- `get_path_groups`
- `get_pins`
- `get_ports`

Examples

- To return a collection of all input ports (same as running `[all_inputs]`), use the following command:
  ```
  get_ports -filter "direction==in"
  ```
- To return a collection of pins that match `*/D`, use the following command:

```
get_pins -hier -filter "hierarchical_name =~ */D"
```

- To return a collection of the libraries a cell belongs to, use the following command:
  ```
  get_lib_cells INVXL
  ```

property Commands

The report_object_property_names command lists the properties for the different object types. This command, without arguments, will list all the properties for all object types. Use the `-type` option to list properties for a specific object type. Legal option types are: `cell`, `clock`, `design`, `lib`, `lib_cell`, `lib_pin`, `lib_timing_arc`, `net`, `path_group`, `pin`, `port`, `timing_arc`, `timing_path`, or `timing_point`.

Examples

- The get_property command returns the value of the specified property. To report the clock period of clk1, use the following command:
  ```
  get_property [get_clocks clk1] period
  ```
- The report_property command returns the properties and their values for a specified object. To report the properties for clock clk1, use the following command:
  ```
  report_property [get_clocks clk1]
  ```
- For timing related properties such as properties for timing arcs, paths, and points run `report_timing -collection`. This will return the results in a collection of timing paths.
  - To report timing and store the top 10 paths as a collection in the variable `$paths`, use the following command:
    ```
    set paths [report_timing -collection -max_paths 10]
    ```
  - To filter `$paths` to include only those paths with slack between -0.806ns and 0ns, use the following command:
    ```
    set newPaths [filter_collection $paths {(slack > -0.806 && slack < 0.0)}]
    ```

collection Commands

Once you have the collection defined you can manipulate, filter, cycle through, and query it using the collection related commands:

- `add_to_collection`
- `append_to_collection`
- `compare_collections`
- `copy_collection`
- `filter_collection`
- `foreach_in_collection`
- `index_collection`

- `query_objects`
- `remove_from_collection`
- `sizeof_collection`
- `sort_collection`

Examples

- To list the contents of a collection, use the following command:

  ```
  query_objects $collectionName
  ```

- To cycle through the items of a collection, use the following command:

  ```
  foreach_in_collection iCell [get_cells *] {
  Puts "Cell = [get_property $iCell hierarchical_name]"
  }
  ```

add_to_collection

```
add_to_collection
base_collection
second_collection_or_list
[-unique]
```

Creates a new collection by adding objects from one collection to another collection. The following rules apply:

- If all of the objects in the base collection are of the same type, only the objects in the second collection that are the same type as those in the base collection are added to the new collection. If you specify an object list instead, all of the objects in the list must be the same type as those in the base collection.
- If the base collection contains different types of objects, all objects in the second collection are added. In this situation, you cannot specify an object list.

If the software finds no objects to add, an empty collection is returned.

Parameters

base_collection	Specifies the base collection of objects. Objects from the second collection or object list are added to this collection to create the new collection.
second_collection_or_list	Specifies the second object collection, or a list of objects.
-unique	Removes any duplicate objects from the new collection.

Related Information

- append_to_collection
- compare_collections
- copy_collection
- filter_collection

all_clocks

`all_clocks`

Creates a collection of all the clocks in the current design. If no clocks were defined, an empty string `""` is returned.

To filter clocks, use the `get_clocks` command to create a collection of clocks matching a specific pattern. Assign these clocks to a variable or pass them using another command.

Examples

- The following example applies the `set_propagated_clock` command to all the clocks in the design:

  ```
  set_propagated_clock [all_clocks]
  ```

Related Information

- `create_clock`
- `get_clocks`
- `set_propagated_clock`
- `all_inputs`

all_connected

`all_connected [-leaf] obj`

Returns a collection of all of the objects connected to the specified net, pin, or port.

Parameters

`-leaf`	Returns only the flattened connection points (leaf pins) for the specified net. *Default:* Returns all hierarchical ports connected to the specified net
obj	Specifies the net, pin, or port for which to return the collection of connected objects. You can specify an object name, or a single-object collection.

Examples

- The following command returns a collection of all of the objects connected to the pin `b1_Sin1`:
  all_connected Sin1

  The software returns the following information:

  Sin1
- The following command returns a collection of all of the objects connected to a pin contained in the single collection object i_block1/b1_Sin1:
  all_connected [get_pins {i_block1/b1_Sin1}]

  The software returns the following information:

  Sin1
- The following command returns a collection of all of the hierarchical ports connected to the net `net_1`:
  all_connected [get_nets net_1]

  The software returns the following information:

  TL2/I2_in TL1/I1_out
- The following command returns a collection that contains only the flattened connection points for the net `net_1`:
  all_connected -leaf [get_nets net_1]

  The software returns the following information:

  TL2/L1/RX/D TL1/RX/Q

all_fanin

```
all_fanin
[-help]
[-hpin]
[-only_cells]
[-startpoints_only]
-to {collection | object_list}
[-trace_through {case_disable |user_disable | all | clocks | loop_snipped}]
[-view view_name]
[-levels value | -pin_levels value]
[ > | >> ]
```

Returns a collection of pins, ports, or cells that exist in the fanin cone of the specified objects.

Note: The `all_fanin` command does not work on netlist traversal but on timing graph traversal.

Parameters

`-help`	Prints out the command usage.
`> | >>`	Stores the generated collection in the specified file. You can add the `.gz` extension to the file name to generate a compressed report. **Note:** The file name must be the last parameter specified.
`-hpin`	Returns hierarchical pins along with the flat pins.
`-levels` *value*	Specifies the number of gate levels to go back from a specific pin object to create the collection.
`-only_cells`	Returns a collection only of the cells in the fanin zone.
`-pin_levels` *value*	Specifies the number delay arcs to go back from a specific pin object to create the collection.
`-startpoints_only`	Returns a collection only of valid start points in the fanin zone.
`-trace_through {case_disable |user_disable | all | clocks | loop_snipped}`	You can specify one of the following options: • `case_disable`: Allows traversal of arcs disabled using `set_case_analysis` command. Arcs disabled using `set_disable_timing` are not traversed. • `user_disable`: Allows traversal of arcs disabled using `set_disable_timing` command, but arcs disabled using `set_case_analysis` command are not traversed. • `all`: Allows traversal through all the arcs • `clocks`: Allows traversal through pins for which clock constraints (such as, `create_clock`, `create_generated_clock`) are applied. The pins that are disabled using the `set_case_analysis` and `set_disable_timing` commands are not traversed. • `loop_snipped`: Allows trace through the loop snipped arcs.
`-to {`*collection | object_list*`}`	Specifies the objects for which to return the collection. You can specify a list of objects, or a collection of objects.
`-view` *view_name*	Specifies the view name for which fan cone traversal is to be performed.

Examples

- The following command returns the collection of cells in the fanin zone of the output port OUT1:
  ```
  all_fanin -to OUT1 -only_cells
  ```
- The following command returns all the valid startpoints (sequential elements and/or input ports) in the fanin zone of OUT1, allowing the traversal through arcs disabled by `set_case_analysis` and `set_disable_timing`:
  ```
  all_fanin -to OUT1 -startpoints_only -trace_through all
  ```

Related Information

- `all_fanout`

all_fanout

```
all_fanout
[-help]
[-endpoints_only]
[-hpin]
[-only_cells]
[-trace_through {case_disable | user_disable | all | clocks | loop_snipped}]
[-view view_name]
[-levels value | -pin_levels value]
{-from {collection | object_list} }
[ > | >> ]
```

Returns a collection of pins, ports, or cells that exist in the fanout cone of the specified objects.

Note: The `all_fanout` command does not work on netlist traversal but on timing graph traversal.

Parameters

`-help`	Prints out the command usage.
`-endpoints_only`	Returns a collection only of valid end points in the fanout zone.
`-hpin`	Returns hierarchical pins along with the flat pins.
`> | >>`	Stores the generated collection in the specified file. You can add the `.gz` extension to the file name to generate a compressed report. **Note:** The file name must be the last parameter specified.
`-from {`*`collection | object_list`*`}`	Specifies the objects for which to return the collection. You can specify a list of objects, or a collection of objects.
`-levels` *`value`*	Specifies the number of gate levels to go forward from a specific pin object to create the collection.
`-only_cells`	Returns a collection only of the cells in the fanout zone.
`-pin_levels` *`value`*	Specifies the number delay arcs to go forward from a specific pin object to create the collection.
`-trace_through {case_disable |user_disable | all | clocks | loop_snipped}`	Allows traversal through disabled arcs. You can specify one of the following options: • `case_disable`: Arcs which are disabled using the `set_case_analysis` command and through physical constant are traversed. The arcs disabled using `set_disable_timing` are not traversed. • `user_disable`: Arcs disabled using `set_disable_timing` command are traversed but those disabled using `set_case_analysis` command are not traversed. • `all`: Enables trace through all arcs • `clocks`: Allows traversal through pins for which clock constraints (such as, `create_clock`, `create_generated_clock`) are applied. The pins that are disabled using the `set_case_analysis` and `set_disable_timing` commands are not traversed. • `loop_snipped`: Allows trace through the loop snipped arcs.
`-view` *`view_name`*	Specifies the view name for which fan cone traversal is to be performed.

Examples

- The following command returns the collection of cells in the fanout zone of the input port IN1:
  ```
  all_fanout -from IN1 -only_cells
  ```
- The following command returns all the valid endpoints (sequential elements and/oroutput ports) in the fanout zone of IN1, allowing the traversal through arcs disabled by `set_case_analysis` and `set_disable_timing`:
  ```
  all_fanout -from IN1 -endpoints_only -trace_through all
  ```

Related Information

- `all_fanin`

all_inputs

```
all_inputs
[-clock list_of_clocks]
[-no_clocks]
[-edge_triggered]
[-level_sensitive]
```

Creates a collection of all the input ports in the current design. To filter inputs, use the `get_ports` command to create a collection of ports matching certain criteria. Assign these ports to a variable or pass them using another command. You can limit the contents of the collection to ports with input delay relative to a specific clock.

Parameters

`-help`	Prints out the command usage.
`-clock` *`list_of_clocks`*	Considers only input ports with input delay relative to the specified list of clocks.
`-edge_triggered`	Returns a collection of input ports with the `set_input_delay -clock` constraint.
`-level_sensitive`	Returns a collection of input ports with the `set_input_delay -level_sensitive` constraint.
`-no_clocks`	Returns primary input ports that do not have create_clock or create_generated_clock constraints.

Example

- The following command returns the names all of the input ports with input delay relative to an `ideal_clk`:
  all_inputs -clock ideal_clk

Related Information

- `get_ports`
- `report_ports`
- `set_input_delay`

all_instances

```
all_instances
obj
[-hierarchical]
```

Returns a collection of all instances available in a design, or all instances of a specified library cell existing in the current design.

You cannot use the `all_instances` command on the top level of the design (`all_instances [get_design top]`).

However, you can use the command for hierarchical modules inside the top level. For example, if `SUB` is a hierarchical module in the top level of the design, you can specify the following command to return all instances of that module: `all_instances [get_design SUB]`.

Parameters

`-hierarchical`	Specifies that the patterns should be matched hierarchically. If the hierarchical name of an instance at a particular hierarchical level matches the patterns, it is included in the result.
obj	Specifies the name of a design or a library cell in the current design. You can specify a pattern for the design or library cell name, or a single collection of a design or library cell.

all_outputs

```
all_outputs
[-clock list_of_clocks
[-edge_triggered]
[-level_sensitive]
```

Creates a collection of all output ports in the current design. To filter outputs, use the `get_ports` command to create a collection of ports matching certain criteria. Assign these ports to a variable or pass them using another command. You can limit the contents of the collection to ports with output delay relative to a specific clock.

Parameters

`-help`	Prints out the command usage.
`-clock` *`list_of_clocks`*	Considers only output ports with an output delay relative to a specific list of clocks.
`-edge_triggered`	Returns a collection of output ports with the `set_output_delay -clock` constraint.
`-level_sensitive`	Returns a collection of output ports with the `set_output_delay -level_sensitive` constraint.

Example

- The following command returns the names all of the output ports with an output delay relative to an `ideal_clk`:

  ```
  all_outputs -clock ideal_clk
  ```

Related Information

- `get_ports`
- `report_ports`
- `set_output_delay`

all_registers

```
all_registers
[-clock {clock_list}]
[-cells]
[-rise_clock {clock_list}]
[-fall_clock {clock_list}]
[-flops | -edge_triggered]
[-no_hierarchy]
[-latches | -level_sensitive]
[-macros]
[-master_slave]
[-data_pins]
[-clock_pins]
[-output_pins]
[-async_pins]
[-slave_clock_pins]
```

Returns a collection of instances and pins of flip-flops, latches, or sequential macros in the design. A cell is considered sequential if the library cell defines trigger or timing check arcs.

Note: The following options can be used only after building the timing graph:

- `-clock`
- `-rise_clock`
- `-fall_clock`

Parameters

`-async_pins`	Returns a collection of async preset or clear pins.
`-cells`	Creates a collection of cells. This is the default option.
`-clock {clock_list}`	Specifies the name of a clock or a list of clocks. The `all_registers` command creates a collection of registers clocked by the specified *`clock_list`*.
`-clock_pins`	Returns a collection of register clock pins. You can use other parameters to further filter this collection.
`-data_pins`	Returns a collection of register data input pins. You can use other parameters to further filter this collection.
`-fall_clock {clock_list}`	Specifies the name of a clock or a list of clocks. The `all_registers` command creates a collection of registers opened by the falling edge of the clocks specified in the *`clock_list`*.

`-flops	-edge_triggered`	Returns a collection of edge-triggered sequential instances. Both the `-flops` and `-edge_triggered` options return the same collection. The `-flops` option is provided for backward compatibility.
`-latches	-level_sensitive`	Returns a collection of level-sensitive sequential instances. Both `-latches` and `-level_sensitive` options return the same collection. The `-latches` option is provided for backward compatibility.
`-macros`	Returns sequential instances that are neither flip-flops nor latches.
`-master_slave`	Returns a collection of master-slave register cells.
`-no_hierarchy`	Searches the current instance only; does not descend the hierarchy.
`-output_pins`	Returns a collection of register output pins.
`-rise_clock` {*`clock_list`*}	Specifies the name of a clock or a list of clocks. The `all_registers` command creates a collection of registers opened by the rising edge of the clocks specified in the *`clock_list`*.
`-slave_clock_pins`	Returns a collection of slave clock pins of the master-slave registers. You can specify the slave clock pins as `clocked_on_also` in the library.

Example

- The following command reports the10 worst paths going to registers in the design: report_timing -to [all_registers] -max_paths 10
- the following script using the `all_registers` command along with other Innovus commands can be used to get number of flops in the design:
  set flopName [all_registers -flops]

  set flopCount [sizeof_collection $flopName]

  puts "Number of flops in the design is $flopCount"

Related Information

- `get_cells`
- `get_pins`

append_to_collection

```
append_to_collection
var_name
second_collection_or_list
[-unique]
```

Appends objects from one collection to another collection. The following rules apply:

- If all of the objects in the base collection are of the same type, only the objects in the second collection that are the same type as those in the base collection are appended to the base collection.
- If the base collection contains different types of objects, all objects in the second collection are appended to it. In this situation, you cannot specify an object list.

If the software finds no objects to add, an empty collection is returned.

Parameters

var_name	Specifies a pointer to the base collection of objects. Objects in the second collection or object list are appended to this collection. If you specify a pointer that does not refer to an existing base collection, the software creates a new collection that uses the pointer as a reference.
second_collection_or_list	Specifies the second collection of objects, or an object list.
-unique	Retains any duplicate objects that already exist in the base collection, but does not append duplicate objects from the second collection to the base collection.

Examples

- The following command appends the nets in the specified collection to the collection `newVar`:
  ```
  append_to_collection newVar [get_nets {CLK1}]
  ```
- The following command appends the specified nets to the collection `newVar`:
  ```
  append_to_collection newVar {Sin1 CLK1 Sout1}
  ```
- You can use the following commands to create another collection:
  ```
  tempus 6> set base_clock [get_clocks sbw_clk]
  0xe50
  tempus 7> set top_clock [get_clocks aclk]
  0xe51
  tempus 8> set base_top_clock ""
  tempus 13> append_to_collection base_top_clock $base_clock
  0xe54
  tempus 14> append_to_collection base_top_clock $top_clock
  0xe54
  ```

```
tempus 15> get_clocks $base_top_clock
sbw_clk aclk
0xe55
tempus 16> get_clocks $base_clock
sbw_clk
0xe56
tempus 17> get_clocks $top_clock
aclk
0xe57
```

Note: If a new collection variable is defined with an existing collection, then both point to the same object set. Any modifications made into either collection will be visible in both cases - because they reference each other.

Related Information

- remove_from_collection
- compare_collections
- copy_collection
- filter_collection

compare_collections

`compare_collections` *`collection1 collection2`* `[-order_dependent]`

Compares two collections, and returns a value of `0` if all of the objects contained in both collections are the same. If the objects contained in both collections are different, the software returns a value of `1`.

Parameters

`collection1`	Specifies the first collection.
`collection2`	Specifies the second collection.
`-order_dependent`	Specifies that the order of the objects in both collections must be the same for the collections to be considered the same.

Related Information

- `remove_from_collection`
- `copy_collection`
- `filter_collection`

copy_collection

`copy_collection` *`base_collection`*

Returns a collection that is an exact copy of the specified base collection.

Parameter

`base_collection`	Specifies the base collection of objects.

Related Information

- `add_to_collection`
- `append_to_collection`
- `remove_from_collection`
- `compare_collections`
- `filter_collection`

define_property

```
define_property
-object_type string
-type data_type
[-lower_bound val]
[-upper_bound val]
[-string_values {values}]
property_name
```

Defines a new property.

Design objects can be associated with user-defined commands that can be used for analysis and debugging purposes.

You can use the `define_property` command if a define statement is not used for the specified property value(s). To read property values from the library, the command must be issued before the timing library is read.

Parameters

`-lower_bound val`	Specifies a lower range - integer, float, or double.
`-object_type <string>`	Specifies the type of object on which the property is to be defined.
property_name	Specifies the name of the property.
`-string_values values`	Specifies the values that can be defined for the property.
`-type data_type`	Specifies the type of property.
`-upper_bound val`	Specifies a higher range - integer, float, or double.

Examples

- The following example defines property having object type `pin` and type of property `int`:
  ```
  define_property -object_type pin -type int slk
  ```
- The following example defines a property on a library arc:
  ```
  define_property -object_type lib_arc -type int libArcProperty
  ```
- The following example defines `aocv_weight` property on a library arc:
  ```
  define_property -type float -object_type lib_arc aocv_weight
  ```

Related Information

- get_property
- set_property
- reset_property

filter_collection

```
filter_collection
[-help]
base_collection
{filter_expression}
[-nocase]
[-quiet]
[-regexp]
```

Returns a collection of objects that were filtered from the specified collection of objects based on user-specified criteria (*filter_expression*). The following rules apply:

- The *filter_expression* can be created by combining several object properties using the following relational and logical operators: >, <, ==, !=, <=, >=, &&, ||, =~, !~, AND, OR. You also can use parentheses to create expressions.
- You can use the defined and undefined existence operators to determine whether an attribute is defined for an object.
- All design object properties supported through the get_property command can be used for filtering the base collection.

Note: You also can perform filtering operations by specifying the -filter parameter with the get_* commands.

Parameters

-help	Prints out the command usage.
base_collection	Specifies the collection of objects from which to filter objects for the new collection.
filter_expression	Specifies the criteria with which to filter objects. The software adds any objects in the base collection that match *filter_expression* to the new collection.
-nocase	Specifies that pattern matching is not case sensitive. **Note:** You must specify -regexp in order to use this parameter.
-quiet	Suppresses all the error and warning messages that are generated when the filter_collection command is run.
-regexp	Treats the specified patterns as regular expression patterns. *Default:* Treats the specified pattern as a wildcard.

Examples

- The following command returns a single-object collection containing the pin with the hierarchical name lat_3/Q:
  ```
  filter_collection [get_pins] {hierarchical_name == lat_3/Q}
  ```

- The following command returns a collection of pins that have a maximum slew that is greater than 0.1:

  ```
  filter_collection [get_pins] {slew_max_rise > 0.1}
  ```

- The following commands create a new collection called `newPaths` that contains all of the paths from the collection `paths` that have a slack value that is greater than -0.806 and less than 0.0:

  ```
  set paths [report_timing -collection -max_paths 10]
  set newPaths [filter_collection $paths {(slack > -0.806 && slack < 0.0)}]
  ```

- The following example applies some values to a few register clock pins:

  ```
  set_property [get_pin hier1/reg1/clock] useful_skew 1.5
  set_property [get_pin hier1/hier2/reg2/clock] useful_skew 2.5
  ```

  In the script, you can look for pin objects that have (or does not have) this attributed applied:

  ```
  find clock pins without useful skew
  set pins_without_useful [filter_collection [all_registers -clock_pins] \
  "undefined(useful_skew)"]
  count the number of registers with useful skew applied
  set pins_with_useful [filter_collection [all_registers -clock_pins] \
  "defined(useful_skew)"]
  ```

Related Information

- add_to_collection
- append_to_collection
- remove_from_collection
- compare_collections

foreach_in_collection

`foreach_in_collection` *`var_name collection body`*

Iterates through all objects in the specified collection and executes the commands in the specified script.

Parameters

`body`	Specifies the script of commands to execute during each iteration.
`collection`	Specifies the collection of objects through which to iterate.
`var_name`	Specifies the iterator variable. During each iteration, *`var_name`* is set to a collection of exactly one object. Any command that accepts collections can accept *`var_name`* as a value.

Examples

- A script with the following commands would print out the hierarchical names of all of the cells in the design:

```
foreach_in_collection iCell [get_cells *] {
puts "Cell = [get_property $iCell hierarchical_name]"

    }
    Cell = ck_0
    Cell = ck_1
    Cell = ck_2
    Cell = ck_4
```

- The following example shows a more complex use of the `foreach_in_collection` command.

```
set cellCollection [get_cells temp1_reg]
set libCellCollection [get_lib_cells -of_objects $cellCollection]
set libArc [get_lib_arcs -of_objects $libCellCollection]

foreach_in_collection arc $libArc {
set fpin [get_property $arc from_lib_pin]

set tpin [get_property $arc to_lib_pin]
set sense [get_property $arc timing_type]

set from_pin_name [get_property $fpin name]
set to_pin_name [get_property $tpin name]

puts [format "%25s -> %25s %-20s" $from_pin_name $to_pin_name $sense]

}
CK -> D           setup_rising
CK -> D           hold_rising
CK -> CK          minimum_period
CK -> CK          minimum_period
CK -> CK          min_pulse_width
```

```
foreach_in_collection.txt

CK -> CK          min_pulse_width
RN -> RN          min_pulse_width
CK -> RN          recovery_rising
CK -> RN          removal_rising
CK -> Q           falling_edge
RN -> Q           clear
```

Related Information

- add_to_collection
- compare_collections
- copy_collection
- filter_collection

get_activity

```
get_activity
[-help]
[-out_file filename]
[-net netname]
[-pin pinname]
[-port portname]
[-summary]
[-tcl_list]
[-list_of_nets_based_on_driver {primary_inputs | sequential | combinational | memory | ICG | black_box}]
[-list_of_nets_based_on_source_of_activity_info
{activity_file|set_switching_activity|propagated|default}]
[-report_average_switching_activity]
[-report_cell_group_activity_summary]
[-inst inst_name]
```

Determines the source of activity for the specified nets/pins/ports. The power analysis engine has various sources from where activities can be annotated from, such as switching activity specification (input/sequential/global/clock gate), clock definitions/constants through SDC/TWF, activity files (VCD/TCF/SAF), user defined activity and/or propagated values.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `get_activity` parameter. For a detailed description of the command and all of its parameters, use the man command: `man get_activity`
`-inst inst_name`	
	Specifies the top/hierarchy instance or the leaf level instance name for which average activity is to be reported. *Data_type*: string, optional
`-list_of_nets_based_on_driver {primary_inputs | sequential | combinational | memory | ICG | black_box}`	
	Specifies to output the list of nets based on the specified net driver. You can specify one or more net drivers at a time, as shown below: `get_activity -list_of_nets_based_on_driver primary_inputs|combinational| sequential | ICG | memory | black_box` To specify multiple drivers, you can use **/** (means 'or') or **&** (means 'and') between two drivers, without a space. *Data_type*: string, optional
`-list_of_nets_based_on_source_of_activity_info {activity_file|set_switching_activity|propagated|default}`	

	Specifies to output the list of nets based on the specified activity source information. *Data_type*: string, optional
`-net` *netname*	Specifies the name of the net. **Note:** You can use the `get_nets` command to determine the activity source for a collection of nets. *Data_type*: string, optional
`-out_file` *filename*	Specifies the output file into which the activity sources are to be written. The format of the report is: <*net_name/pin_name*> <*activity_source*> <*duty_cycle*> <*transition_density*> For more information, refer to the "Static Power Analysis Overview" section of the Power Analysis and Reports chapter in *Innovus System User Guide*. *Data_type*: string, optional
`-pin` *pinname*	Specifies the name of the pin. **Note:** You can use the `get_pins` command to determine the activity source for a collection of pins. *Data_type*: string, optional
`-port` *portname*	Specifies the name of the port. **Note:** You can use the `get_ports` command to determine the activity source for a collection of ports. *Data_type*: string, optional
`-report_average_switching_activity`	
	Specifies to report the average activity of the given instance/hierarchy/top-level. *Data_type*: bool, optional
`-report_cell_group_activity_summary`	
	Specifies to give a summary of cell group activity. *Data_type*: bool, optional

`-summary`	Generates a detailed activity annotation report of the specified nets, pins, and ports. This report contains the following information: • Primary Inputs • Sequential Outputs • Memory/Macro Outputs • Tristate Outputs **Note:** Use the `get_activity -summary` command to report activity annotation summary without printing information for nets/pins/ports. *Data_type*: bool, optional
`-tcl_list`	Produces the report in the Tcl list format instead of a tabular format. The `-tcl_list` and `-out_file` parameters are mutually exclusive; you cannot specify them together. *Data_type*: bool, optional

Examples

- The following example returns the activity sources of all nets that match the specified pattern and writes the output information to a file named `nets.rpt`:

  ```
  get_activity -net [get_nets n*] -out_file nets.rpt
  ```

  The following is the example output of the nets.rpt file:

  ```
  int3 activity_file 0.5 5e+07
  int2 sequential_activity 0.5 2.5e+07
  int1 user_defined_activity 0.5 1e+07
  nt3 global_activity 0.5 5e+07
  nt3 input_activity 0.5 5e+07
  int5 clock_gate_activity 0.5 2.5e+07
  int1 user_defined_activity 0.5 1e+07out
  sequential_activity 0.5 2.5e+07
  clk sdc/twf 0.5 5e+08
  c_int2 propagation 0.5 5e+08
  ```

- The following example returns the activity source of the pin `i_4/A1` and writes the output information to a power report file named `pin.rpt`:

  ```
  get_activity -pin i_4/A1 -out_file pin.rpt
  ```

- The following command report the average activity of all nets:

  ```
  get_activity -report_average_switching_activity
  ```

  The following is the example output:

  ```
  Avg Switching activity of all nets 4.727e+17
  ```

- The following command gives a summary of cell group activity:
  ```
  get_activity -report_cell_group_activity_summary
  ```
  The following is the example output:
  ```
  Group Activity Summary Report:
  Source ->                    activity file            set_switching_activity
  propagated              default                  Total
  All nets                     2                        1
  2                       5                        10
  Break-up based on net driver:
  Primary Input                2(40%)                   0(0%)
  0(0%)                   3(60%)                   5
  Sequential                   0(0%)                    1(33.33%)
  0(0%)                   2(66.67%)                3
  Combinational                0(0%)                    0(0%)
  2(100%)                 0(0%)                    2
  Memory                       0(0%)                    0(0%)
  0(0%)                   0(0%)                    0
  ICG                          0(0%)                    0(0%)
  0(0%)                   0(0%)                    0
  Black Box                    0(0%)                    0(0%)
  0(0%)                   0(0%)                    0
  ```
- The following command gives the list of nets based on primary inputs:
  ```
  get_activity -list_of_nets_based_on_driver primary_inputs
  ```
  The following is the example output:
  ```
  D2 D1 CP2 CP1 dummy0
  ```
- The following command gives the list of nets based on primary inputs or sequential driver:
  ```
  get_activity -list_of_nets_based_on_driver primary_inputs|sequential
  ```
  The following is the example output:
  ```
  Q2 Q1 D2 D1 CP2 CP1 n1 dummy0
  ```
- The following command gives the list of nets based on primary inputs or sequential driver for the instance `ff1b`:
  ```
  get_activity -list_of_nets_based_on_driver primary_inputs|sequential -instance ff1b
  ```
- The following command gives the list of nets based on the propagated source for the instance `i2`:
  ```
  get_activity -list_of_nets_based_on_source_of_activity_info propagated -instance i2
  ```
- The following command gives a summary of cell group activity for the instance `i2`:
  ```
  get_activity -instance i2 -report_cell_group_activity_summary
  ```
- The following command gives the list of nets based on the default activity:
  ```
  get_activity -list_of_nets_based_on_source_of_activity default
  ```

  The following is the example output:
  ```
  rst shutdown isolate coef_ld start bypass test_mode se sdi Xn_in[3] Xn_in[2] Xn_in[1] Xn_in[0]
  Yn_in[15] Yn_in[14] Yn_in[13] Yn_in[12] Yn_in[11] Yn_in[10] Yn_in[9] Yn_in[8] Yn_in[7] Yn_in[6]
  ```
- The following command gives the list of nets based on the activity file:
  ```
  get_activity -list_of_nets_based_on_source_of_activity activity_file
  ```

  The following is the example output:

```
ref_clk test_clk pll_clk
```

- The following command gives the list of nets based on the switching activity:
  ```
  get_activity -list_of_nets_based_on_source_of_activity set_switching_activity
  ```

  The following is the example output:
  ```
  rst ref_clk shutdown isolate coef_ld start bypass test_clk test_mode se sdi sdo Xn_in[3] Xn_in[2]
  Xn_in[1] Xn_in[0] Yn_in[15] Yn_in[14] Yn_in[13]
  ```

Related Information

- power Category Attributes
- `report_power`
- "Static Power, IRdrop and EM Analysis" in the *Voltus Stylus Common UI User Guide*
- "Dynamic Power and IRDrop Analysis" in the *Voltus Stylus Common UI User Guide*

get_arcs

```
get_arcs
{ [-to to_list] [-from from_list]
 | -of_objects object_list}
[-filter expr]
[-quiet]
```

Creates a collection of timing arcs. Assign these arcs to a variable, or pass them to another command. If no objects match the criteria, an empty collection is returned.

Note: The timing graph must be built before you can use this command. A warning message is issued indicating that the timing graph is not generated.

Parameters

`-filter` *expr*	Filters a collection based on the specified expression. For any arcs that match the pattern, the software evaluates the expression based on the attribute of the arc.
`-from` *from_list*	Adds all timing arcs that fan in directly to the specified pins or ports to the collection. You can specify a list of pins or a list of ports, but you cannot specify a list that contains both pins and ports. You can specify a pattern for the pin or port name, a collection of pins or ports, or a pattern and a collection. **Note:** You cannot specify this parameter with the `-of_objects` parameter.
`-of_objects` *object_list*	Adds all timing arcs that are associated with the specified cells or nets to the collection. You can specify a list of cells or a list of nets, but you cannot specify a list that contains both cells and nets. You can specify a pattern for the cell or net name, a collection of cells or nets, or a pattern and a collection. **Note:** You cannot specify this parameter with the `-to` or `-from` parameters.
`-quiet`	Suppresses all error and warning messages generated when the `get_arcs` command is run.
`-to` *to_list*	Adds all timing arcs that fan out directly from the specified list of pins or ports to the collection. You can specify a list of pins or a list of ports, but you cannot specify a list that contains both pins and ports. You can specify a pattern for the pin or port name, a collection of pins or ports, or a pattern and a collection. **Note:** You cannot specify this parameter with the `-of_objects` parameter.

Examples

- The following command returns a collection of the timing arcs fan in directly to the pin `dff1/CP`:
  get_arcs -from dff1/CP
- The following command creates a collection of the timing arcs that are associated with the nets in the specified

collection:
get_arcs -of_objects [get_nets {n1}]

get_cells

```
get_cells
[-help]
[-filter expr]
[-hierarchical]
[-hsc char]
[-leaf]
[-nocase]
[-quiet]
[-regexp]
[patterns | -of_objects object_list]
```

Creates a collection of instances in the current design whose name matches the supplied pattern list. Assign this collection to a variable or pass it as an argument in another command.

Parameters

`-filter` *expr*	Filters the collection with the specified expression. For any cells that match the pattern, the software evaluates the expression based on the attribute of the cell. You can filter cells using any of the cell properties supported through the `get_property` command.
`-hierarchical`	Specifies that the patterns should be matched hierarchically. If the flat name of an instance at a particular hierarchical level matches the patterns, it is included in the result. For example. if there is top level hierarchical instance `Top` and it contains another sub-instance called `FF1`, then the following command can be used to get the instance `Top/FF1` (since the given pattern will be matched with instance flat name `FF1`): get_cells -hier FF* However, the following command will display an error: get_cells -hier Top/FF*
`-hsc` *char*	Specifies the hierarchical delimiter for patterns. For example, if the hierarchical delimiter is `@`, the `sub/I1@I2` pattern matches instance `I2` within the hierarchy limited at `sub/I1`.
`-leaf`	Returns leaf cells only. When the `-leaf` parameter is not specified, the `get_cells` command returns the hierarchical instances present on a net along with the flat instances at the same hierarchical level as the specified object. **Note:** The `-leaf` parameter can only be specified with the `-of_objects` parameter.
`-nocase`	Specifies that pattern matching is not case sensitive. **Note:** You must specify `-regexp` in order to use this parameter.

`-of_objects` *object_list*	Creates a collection of all cells associated with the specified pins, nets, library cells, or design cells.
	You can specify a list of pins or a list of nets, but you cannot specify a list that contains both pins and nets. You can specify patterns for the pin or net names, a collection of pins or nets, or a combination of patterns and a collection.
	The `-of_objects` parameter supports `get_designs` and `get_lib_cells` as valid collection objects.
	The use model is as follows:
	`get_cells -of_objects [get_designs` *object_list*`]`
	`get_cells -of_objects [get_lib_cells` *object_list*`]`

patterns	Specifies patterns for instance names. Patterns include the `*` and `?` wildcard characters and collections of instances. The `get_cells` command returns a collection of instances that match the patterns. If no such instance is found, an empty collection is returned.
`-quiet`	Suppresses all error and warning messages generated when the `get_cells` command is run.

`-regexp`	Treats the specified patterns as a regular expression patterns.
	When this parameter is specified, the given patterns `start` and `end` are assumed to be matching with the start and end part of the complete object name.
	For example, the following can be used to retrieve the hierarchical instance `Top/block1/block2/FF1`:
	`get_cells Top/*/*/FF1`
	`get_cells -regexp Top/.*/FF1`
	However, the following command will display an error :
	`get_cells Top/*/FF1`
	If this parameter is specified with `-hier` parameter, the given pattern will be matched with the complete hierarchical name of objects instead of only the flat name and all the successful matches are included in the result.
	For example, if there is a top level hierarchical instance `Top` and it contains another sub-instance called `FF1`, then either of these can be used to retrieve `Top/FF1` instance:
	`get_cells -hier FF1`
	`get_cells -hier -regexp Top/FF1`
	However, the following commands will display an error :
	`get_cells -hier Top/FF1`
	`get_cells -hier -regexp FF1`

Example

- The following command searches for pins within the hierarchy limited at `B`:

  ```
  get_cells -hsc @ B@*
  ```

- The following command searches for cells with set pattern and filter:

  ```
  get_cells "o*" -filter "@ref_lib_cell_name != FD2 && @is_black_box==false"
  ```

Related Information

- `get_libs`
- `get_lib_cells`
- `get_lib_pins`
- `get_pins`

get_clocks

```
get_clocks
[-filter expr]
[-regexp]
[-nocase]
[-quiet]
[patterns]
```

Creates a collection of clocks whose name matches the supplied pattern list. Assign this collection to a variable or pass it as an argument to another command.

`The get_clocks` command reports all clocks, including inactive clocks.

Parameters

`-help`	Prints out the command usage.
`-filter expr`	Filters the collection with the specified expression. For any clocks that match the pattern, the software evaluates the expression based on the attribute of the clock. You can filter clocks using any of the clock properties supported through the get_property command.
`-nocase`	Specifies that pattern matching is not case sensitive. **Note:** You must specify `-regexp` in order to use this parameter.
patterns	Specifies patterns for clock names. Patterns include the `*` and `?` wildcard characters, and collections of instances and clocks.
`-quiet`	Suppresses all error and warning messages generated when the `get_clocks` command is run.
`-regexp`	Treats the specified patterns as a regular expression patterns. *Default:* Treats the specified pattern as a wild card

Example

- The following example returns all clocks beginning with the name `CK`: get_clocks CK*

Related Commands

- all_clocks
- create_clock
- get_pins
- get_ports
- report_clocks

get_designs

```
get_designs
[-quiet]
patterns
```

Creates a collection of modules and assigns this collection to a variable or pass it as an argument to an another command. The software uses this information during timing analysis.

Parameters

patterns	Specifies patterns for module names. Patterns include the * and ? wildcard characters and collections of modules. The get_designs command returns a collection of modules that match the patterns. If no such module is found, an empty collection is returned.
-quiet	Suppresses all error and warning messages generated when the get_design command is run.

Example

- The following command returns a complete list of modules:

```
get_designs *
```

get_generated_clocks

```
get_generated_clocks
[-filter expr]
[-regexp | -exact]
[-nocase]
[patterns]
```

Creates a collection of generated clocks whose name matches the supplied pattern list. Assign this collection to a variable, or pass it as an argument to another command.

Parameters

`-exact`	Performs exact pattern matching against the object name, instead of wild card or regular expression matching. Use this parameter when the object name includes special characters such as `*`, `?`, and `+`. **Note:** If you specify this parameter, you cannot specify `-regexp`.
`-filter expr`	Filters the collection with the specified expression. For any clocks that match the pattern, the software evaluates the expression based on the attribute of the clock.
`-nocase`	Specifies that pattern matching is not case sensitive.
patterns	Specifies patterns for generated clock names. Patterns include the `*` and `?` wildcard characters, and collections of generated clocks.
`-regexp`	Treats the specified pattern as a regular expression. *Default:* Treats the specified pattern as a wild card **Note:** If you specify this parameter, you cannot specify `-exact`.

Related Information

- `create_generated_clock`

get_lib_arcs

```
get_lib_arcs
{ [-to to_lib_pins] [-from from_lib_pins]
  | -of_objects {lib_cell_list | timing_arcs} }
[-filter expr]
[-quiet]
```

Creates a collection of library timing arcs. Assign these library timing arcs to a variable, or pass them to another command. If no objects match the criteria, an empty collection is returned.

Parameters

`-filter` *expr*	Filters the collection with the specified expression. For any library arcs that match the pattern, the software evaluates the expression based on the attribute of the library arc
`-from` *from_lib_pins*	Creates a collection of all library arcs that originate from the specified library pins. You can specify a pattern for the pin name, a collection of pins, or a pattern and a collection. If you specify this parameter with the `-to` parameter, the software creates a collection of all library arcs that exist between the "to" and "from" library pins. **Note:** You cannot specify this parameter with the `-of_objects` parameter.
`-of_objects` {*lib_cell_list* | *timing_arcs*}	Creates a collection of all library arcs that are associated with the specified library cells or timing arcs. You can specify a list of library cells or a list of timing arcs, but you cannot specify a list that contains both library cells and timing arcs. You can specify a pattern for the library cell or timing arc name, a collection of library cells or timing arcs, or a pattern and a collection. **Note:** You cannot specify this parameter with the `-to` or `-from` parameters.
`-quiet`	Suppresses all error and warning messages generated when the `get_lib_arcs` command is run.
`-to` *to_lib_pins*	Creates a collection of all library arcs that terminate at the specified library pins. You can specify a pattern for the pin name, a collection of pins, or a pattern and a collection. If you specify this parameter with the `-from` parameter, the software creates a collection of all library arcs that exist between the "to" and "from" library pins. **Note:** You cannot specify this parameter with the `-of_objects` parameter.

Related Information

- report_lib_arcs
- get_lib_cells
- get_lib_pg_pins
- get_lib_pins
- get_libs

get_lib_cells

```
get_lib_cells
[-filter expr]
[-regexp]
[-nocase]
[-quiet]
{pattern_list | -of_objects object_list}
```

Creates a collection of library cells from the loaded libraries whose name matches the supplied pattern list. Assign these library cells to a variable or pass them to another command. If no objects match the criteria, an empty collection is returned.

Parameters

`-filter expr`	Filters the collection with the specified expression. For any library cells that match the pattern, the software evaluates the expression based on the attribute of the library cell. You can filter library cells using any of the library cell properties supported through the get_property command.
`-nocase`	Specifies that pattern matching is not case sensitive. **Note:** You must specify `-regexp` in order to use this parameter.
`-of_objects object_list`	Creates a collection of all library cells associated with the specified library pins or cells or libs. You can specify a list of library pins or a list of cells or a list of libs, but you cannot specify a list that contains mixed of library pins and cells and libs. You can specify patterns for the library pin or cell or lib names, a collection of library pins or cells or libs, or a combination of patterns and a collection.

pattern_list	Returns only those libraries in the collection whose name matches one or more of the patterns specified in the *pattern_list* argument. The characters ? and * are wild cards. The ? character matches any one character while the * character matches zero or more characters. If a library matches multiple patterns, it appears multiple times in the collection. A pattern can have only one hierarchical separator character. The left part of the pattern is the library name pattern and the right part is the cell name pattern. If there is no separator character, the pattern is compared to cell names in all the libraries. A pattern can have only one hierarchical separator character, dividing the pattern into one or two parts. If the hierarchical separator is /, the alternatives are: • *cell_name_pattern* Selects all cells whose name matches the *cell_name_pattern* from all libraries. • *lib_name_pattern/cell_name_pattern* Selects all cells whose name matches the *cell_name_pattern* from all libraries whose name matches the *lib_name_pattern*.
-quiet	Suppresses all error and warning messages generated when the get_lib_cells command is run.
-regexp	Treats the specified patterns as regular expression patterns. *Default:* Treats the specified pattern as wildcard

Examples

- The following command prints all cells for all libraries: get_lib_cells *
- The following command prints a collection with all cells in the lca500kv lib library: get_lib_cells lca500kv/*
- The following command prints a collection with all cells whose names starts with AN in all libraries starting with lca: get_lib_cells lca*/AN*
- The following command disables delay arcs on the EF11L050032A library cell. The set_disable_cell_timing command effects all instantiations of the specified cell, and lets you disable delay arcs from and to internal pins. set_disable_timing -from WACLK -to RBO1 [get_lib_cells {EF11L050032A}]

Related Information

- get_cells
- get_libs
- get_lib_pins

get_lib_pg_pins

```
get_lib_pg_pins
[-help]
[-filter expr]
{-of_objects object_list}
```

Returns a collection of library-level PG pins from the loaded libraries. An empty collection is returned if no PG pin definitions are found in the loaded library.

Parameters

`-help`	Prints out the command usage.
`-filter expr`	Filters the collection with the specified expression. For any library pg pins that match the specified expression, the software evaluates the expression based on the attribute of the library pg pin. You can filter library pg pins along with the `-of_objects` parameter - using the library pg pin properties of `lib_pg_pin` object type (supported through the `get_property` command).

`-of_objects object_list`	Creates a collection of all library PG pins associated with the specified library cells.
	The `-of_objects` parameter supports `get_lib_cells` as valid objects.
	For example,
	`get_lib_pg_pins -of_objects [get_lib_cells sc9_cmos32lp/DFFQ_X1M] VDD VSS`
	`get_lib_pg_pins *`
	`**ERROR: (TCLCMD-1420): Currently only -of_objects option is supported`

Examples

- The following command gets the collection of all the library-level pg pins of the specified library cell:
  `get_lib_pg_pins -of_objects [get_lib_cells sc_cmoslp/AO22X]`
- The following command filters the collection of all library-level pg pins of the library cell to return only the library pg pins containing the string name defined with the lib_pg_pin name attribute:
  `get_lib_pg_pins -of_objects [get_lib_cells sc_cmoslp/AO22X] -filter "name =~ *VDD*"`

Related Information

- get_lib_cells
- get_lib_pins
- get_libs
- get_property

get_lib_pins

```
get_lib_pins
[-filter expr]
[-regexp]
[-nocase]
[-quiet]
{pattern_list | -of_objects object_list}
```

Creates a collection of library cell pins from the loaded libraries whose name matches the supplied pattern list. Assign these library cell pins to a variable or pass them to another command. If no objects match the criteria, an empty collection is returned.

Parameters

`-filter expr`	Filters the collection with the specified expression. For any library cell pins that match the pattern, the software evaluates the expression based on the attribute of the library cell pin. You can filter library pins using any of the library pin properties supported through the get_property command.
`-nocase`	Specifies that pattern matching is not case sensitive. **Note:** You must specify `-regexp` in order to use this parameter.
`-of_objects object_list`	Creates a collection of all library pins associated with the specified library cells or pins. You can specify a list of cells or a list of pins, but you cannot specify a list that contains both cells and pins. You can specify patterns for the cell or pin names, a collection of cells or pins, or a combination of patterns and a collection.
pattern_list	Returns only those libraries in the collection whose name matches one or more of the patterns specified in the *pattern_list* argument. The characters `?` and `*` are wild cards; The `?` character matches any one character while the `*` character matches zero or more characters. If a library cell matches multiple patterns, it appears multiple times in the collection. A pattern can have only two hierarchical separator characters, dividing the pattern into one, two, or three parts. If the hierarchical separator is `/`, the alternatives are: • *pin_name_pattern* Selects all pins whose name matches the *pin_name_pattern* from all cells in the libraries. • *cell_name_pattern/pin_name_pattern* Selects all pins whose name matches the *pin_name_pattern* from all cells whose name matches the *cell_name_pattern* in all libraries. • *lib_name_pattern/cell_name_pattern/pin_name_pattern* Select all pins whose name matches the *pin_name_pattern* from all cells whose name matches the *cell_name_pattern* from all libraries whose name matches the *lib_name_pattern*.

`-quiet`	Suppresses all error and warning messages generated when the `get_lib_pins` command is run.
`-regexp`	Treats the specified patterns as a regular expression patterns. *Default:* Treats the specified pattern as a wild card

Examples

- The following command returns all pins in all cells for all libraries:
  ```
  get_lib_pins *
  ```
- The following command returns a collection with all pins for cell `AO1` in any library:
  ```
  get_lib_pins AO1/*
  ```
- The following command returns a collection with all pins whose name starts with `Z` or `Q` for the `AO1` cell in any library:
  ```
  get_lib_pins {AO1/Z* AO1/Q*}
  ```
- The following command returns a collection with all pins for the `AO1` cell in the `lca500kv` lib library:
  ```
  get_lib_pins lca500kv/AO1/*
  ```
- The following command returns a collection with all pins starting with `Z` in all cells whose names starts with `AN` in all libraries starting with `lca`:
  ```
  get_lib_pins lca*/AN*/Z*
  ```

Related Information

- `get_libs`
- `get_lib_cells`

get_libs

```
get_libs
[-filter expr]
[-quiet]
[-regexp]
[-nocase]
{-of_objects object_list | pattern_list}
```

Creates a collection of loaded libraries whose name matches those specified in the pattern list. Assign these libraries to a variable or pass them to another command. If no objects match the criteria, an empty collection is returned.

Parameters

`-filter expr`	Filters the collection with the specified expression. For any libraries that match the pattern, the software evaluates the expression based on the attribute of the library. You can filter libraries using any of the library properties supported through the get_property command.
`-nocase`	Specifies that pattern matching is not case sensitive. **Note:** You must specify `-regexp` in order to use this parameter.
`-of_objects object_list`	Creates a collection of libraries that belong to the library cells in a given object list.
pattern_list	Returns only those libraries in the collection whose name matches one or more of the patterns specified in the *pattern_list* argument. The characters `?` and `*` are wild cards; The `?` character matches any one character while the `*` character matches zero or more characters. If a library matches multiple patterns, it appears multiple times in the collection.
`-quiet`	Suppresses all error and warning messages generated when the `get_lib_cells` command is run.
`-regexp`	Treats the specified patterns as a regular expression patterns. *Default:* Treats the specified pattern as a wild card

Examples

- The following command displays a list of loaded libraries starting with a or l: get_libs {a* l*}
- The following command queries all libraries beginning with the letter `1`: get_libs 1*
- The following command creates a collection of libraries that are associated with the library cells in the specified object list:
  get_libs -of_objects [get_lib_cells *]

Related Information

- get_lib_cells
- get_lib_pins

get_nets

```
get_nets
[-hierarchical]
[-hsc char]
[-filter expr]
[-regexp]
[-nocase]
[-quiet]
[patterns | -of_objects object_list]
```

Creates a collection of nets in the current design whose name matches the supplied pattern list. Assign this collection to a variable or pass it as an argument to an another command. If no such net is found, an empty collection is returned.

When you use the `get_nets` command to specify hierarchical nets, the software matches the specified pattern and returns the top level net connected to it.

Parameters

`-filter` *expr*	Filters the collection with the specified expression. For any nets that match the pattern, the software evaluates the expression based on the attribute of the net. You can filter nets using any of the net properties supported through the get_property command.
`-hierarchical`	Specifies that the patterns should match hierarchically. If the hierarchical name of a net at any hierarchical level matches the patterns, it is included in the result.
`-hsc` *char*	Specifies the hierarchical delimiter for patterns.
`-nocase`	Specifies that pattern matching is not case sensitive. **Note:** You must specify `-regexp` in order to use this parameter.
`-of_objects` *object_list*	Creates a collection of all nets associated with the specified cells, pins, or ports. You can specify a list of cells, a list of pins, or a list of ports, but you cannot specify a list that contains cells, pins, and ports together. You can specify patterns for the cell, pin, or port names, a collection of cells, pins, or ports, or a combination of patterns and a collection.
patterns	Specifies patterns for net names. Patterns include the `*` and `?` wildcard characters and collections of nets.
`-quiet`	Suppresses all error and warning messages generated when the `get_nets` command is run.
`-regexp`	Treats the specified patterns as regular expression patterns. *Default:* Treats the specified pattern as wildcard.

Examples

- The following command returns all nets within the hierarchy limited at `A`:

  ```
  get_nets -hsc @ A@*
  ```

- The following command returns all patterns that match hierarchy and identifies nets to be excluded from optimization.

  ```
  get_nets * -hierarchical -filter "@dont_touch== true"
  ```

Related Information

- `get_pins`

get_object_name

`get_object_name` *`collection`*

Returns the name of the object(s) contained in the specified collections.

By default, the command hides escape characters in the output. To enable printing of escape characters in the output, you can set `timing_enable_get_obj_escaped_name_backward_compatible` attribute to `true`.

Parameters

`collection`	Specifies multiple object collection that contain the objects.

Examples

- The following command returns a tcl list of names which can match the collection inputs:

  ```
  get_object_name [get_cells and*]
  ```

- The following command defines a variable called `endpoint_list` to store endpoints with the ten worst slacks:

  ```
  set endpoint_list [get_object_name [get_property \
        [report_timing -collection -max_points 10] capturing_point]]
  ```

Related Information

- `query_objects`

get_path_groups

```
get_path_groups
[-help | {[-nocase ] [ -regexp ] patterns}]
```

Returns a collection of path groups whose names match *patterns*. If no path groups match the criteria, an empty collection is returned.

Parameters

`-nocase`	Specifies that pattern matching is not case sensitive.
patterns	Creates a collection of path groups whose name matches one or more of the specified patterns. Patterns include the `*` and `?` wildcard characters and collections of path groups. If a path group matches multiple patterns, it appears multiple times in the collection.
`-regexp`	Treats the specified patterns as regular expression patterns. *Default:* Treats the specified pattern as wildcard.

Examples

- Iterate through all the path groups in the path group collection: `foreach_in_collection thePathGroup [ get_path_groups ] { ... }`
- Iterate through all path group **DPObjects**: `foreach thePathGroupDPO [ get_db [ get_path_groups ] ] { ... [ get_db $thePathGroupDPO ... ]; ... }`

Related Information

- set_path_group
- group_path
- reset_path_group
- report_path_groups

get_pg_nets

```
get_pg_nets
[-help]
[-filter expr]
[-of_objects object_list]
```

Creates a collection of netlist-level instance PG nets of leaf cells/pins in the current design. An empty collection is returned if no PG net definitions/connections are found in the design netlist.

Parameters

`-help`	Prints out the command usage.
`-filter` *expr*	Specifies the filters for the collection with the specified expression. For any pg nets that match the pattern, the software evaluates the expression based on the attribute of the net. You can filter pg nets along with the `-of_objects` parameter - using any of the pg net properties of object type `pg_net` (supported through the `get_property` command).

`-of_objects` *object_list*	Creates a collection of all the PG nets, or PG nets associated with the specified cells or PG pins.
	You can specify a list of cells or a list of pg pins, but you cannot specify a list that contains cells and pg pins together.
	The `-of_objects` parameter supports `get_cells` and `get_pg_pins` as valid objects.
	For example,
	`get_pg_nets -of_objects [get_cells iA/out_reg] VSS VDD2`
	`get_pg_nets -of_objects [get_pg_pins -of_objects [get_cells iA/out_reg]] VSS VDD2`
	`get_pg_nets *`

Examples

- The following command filters the collection of all pg nets of the cell instance object to return only the pg nets containing the string name defined with the pg_net hierarchical_name attribute:
  ```
  get_pg_nets -of_objects [get_cells iA/out_reg] -filter "hierarchical_name =~ *VDD*"
  ```
- The following command gets the collection of all pg nets of the cell instance object:
  ```
  get_pg_nets -of_objects [get_cells iA/out_reg]
  ```
- The following command gets the collection of all pg nets of the pg_pins of the cell instance object:
  ```
  get_pg_nets -of_objects [get_pg_pins -of_objects [get_cells iA/out_reg]]
  ```

Related Information

- get_property
- get_nets

get_pg_pins

```
get_pg_pins
[-help]
[-filter expr]
[-of_objects object_list]
```

Returns a collection of netlist-level instance PG pins in the current design. An empty collection is returned if no PG pin definitions are found in the design netlist.

Parameters

`-help`	Prints out the command usage.
`-filter` *expr*	Specifies the filters for the collection with the given expression. For any pg pins that match the pattern, the software evaluates the expression based on the attribute of the pin. You can filter pg pins along with the `-of_objects` parameter - using the pg pin properties of object type `pg_pin` (supported through the `get_property` command).

`-of_objects` *object_list*	Specifies the collection of all the pins associated with the specified cells or nets.
	The `-of_objects` parameter supports `get_cells` as valid objects.
	For example:
	`get_pg_pins -of_objects [get_cells iA/out_reg] iA/out_reg/VSS iA/out_reg/VDD`
	`get_pg_pins iA/out_reg/VDD`
	`**ERROR: (TCLCMD-1420): Currently only -of_objects option is supported.`

Examples

- The following command filters the collection of all the pg_pins of the cell instance object to return only the pg pins containing the string name defined with the pg_pin hierarchical_name attribute:
  ```
  get_pg_pins -of_objects [get_cells iA/out_reg] -filter "hierarchical_name =~ *VDD*"
  ```
- The following command gets the collection of all pg pins of the cell instance object:
  ```
  get_pg_pins -of_objects [get_cells iA/out_reg]
  ```

Related Information

- `get_property`
- `get_pg_nets`

get_pins

```
get_pins
[-hierarchical]
[-hsc char]
[-filter expr]
[-leaf]
[-regexp]
[-nocase]
[-quiet]
{patterns | -of_objects object_list}
```

Creates a collection of instance pins whose name matches the supplied pattern list. Assign this collection to a variable or pass it as an argument to another command. If no such pin is found, an empty collection is returned.

Parameters

`-filter` *expr*	Filters the collection with the specified expression. For any pins that match the pattern, the software evaluates the expression based on the attribute of the pin. If the expression is set to true, the pin is included in the result. You can filter pins using any of the pin properties supported through the get_property command.
`-hierarchical`	Specifies that patterns should match hierarchically. If the hierarchical name of a pin at any hierarchical level matches the patterns, it is included in the result.
`-hsc` *char*	Specifies the hierarchical delimiter for patterns.
`-leaf`	Returns leaf pins only. When the `-leaf` parameter is not used, the `get_pins` command returns the hierarchical pins present on a net along with the flat pins at the same hierarchical level as the specified object.
`-nocase`	Specifies that pattern matching is not case sensitive. **Note:** You must specify `-regexp` in order to use this parameter.
`-of_objects` *object_list*	Creates a collection of all pins associated with the specified cells or nets. You can specify a list of cells or a list of nets, but you cannot specify a list that contains both cells and nets. You can specify patterns for the cell or net names, a collection of cells or nets, or a combination of patterns and a collection.
patterns	Specifies patterns for pin names. Patterns include the `*` and `?` wildcard characters and collections of pins.
`-quiet`	Suppresses all error and warning messages generated when the `get_pins` command is run.

`-regexp`	Treats the specified patterns as regular expression patterns. *Default:* Treats the specified pattern as wildcard.

Examples

- The following command returns all instance pins within the hierarchy limited at `A`: get_pins -hsc @ A@*
- The following command filters the collection for `pin_direction` pin attribute. get_pins "in*" -filter "@pin_direction == in"

Related Information

- `create_clock`
- `get_cells`
- `get_ports`

get_ports

```
get_ports
[-filter expr]
[-regexp]
[-nocase]
[-quiet]
[patterns | -of_objects object_list]
```

Creates a collection of ports whose name matches the supplied pattern list. Assign this collection to a variable or pass it as an argument to another command.

Note: The `get_ports` command always creates a collection of top level ports irrrespective of the current instance set using the `current_instance` command.

Parameters

`-filter` *`expr`*	Filters the collection with the specified expression. For any ports that match the pattern, the software evaluates the expression based on the attribute of the port. If the expression is set to true, the port is included in the result. You can filter ports using any of the port properties supported through the `get_property` command.
`-nocase`	Specifies that pattern matching is not case sensitive. **Note:** You must specify `-regexp` in order to use this parameter.
`-of_objects` *`object_list`*	Creates a collection of all ports associated with the specified nets. You can specify patterns for the net names, a collection of nets, or a combination of patterns and a collection.
`patterns`	Specifies patterns for port names. Patterns include the `*` and `?` wildcard characters and collections of ports.The command returns a collection of ports that match the patterns. If no such port is found, an empty collection is returned.
`-quiet`	Suppresses all error and warning messages generated when the `get_ports` command is run.
`-regexp`	Treats the specified patterns as regular expression patterns. *Default:* Treats the specified pattern as wildcard.

Example

- The following example returns all ports beginning with the name `inb`: get_ports inb*
- The following command filters the collection for `port_direction` port attribute. get_ports "mode*" -filter {port_direction == in}

Related Commands

- all_inputs
- all_outputs
- get_clocks
- get_pins

get_power

```
get_power
[-help]
[-out_file filename ]
[-nets nets_list | -pins pins_list | -insts instances_list ]
[-attribute attributes_list ]
[-include_unit]
[-tcl_list ]
[-pg_net {pg_net_name_list | all}]
```

Queries various power related properties of design objects like nets, pins, and instances. Using this command, you can retrieve power attributes, such as internal power, switching power, leakage power, total power, and toggle rate. This command is very useful for writing custom scripts.

Note: This command outputs toggle rates as N.A. if the clock domain information for a given net/pin/instance is not found.

Use this command after running the `report_power` or `restore_power_database` command.

Note: The `-nets`, `-pins`, and `-insts` parameters are mutually exclusive. You must specify only one of the parameters.

Parameters

`-attribute` *attributes_list*	Specifies a list of power attributes. If an attribute is not available for the specified design object, the software returns its value as NA. See List of Power Attributes for a mapping of the list of available attributes with the design objects. This is a required parameter. *Data_type*: string, optional
`-help`	Outputs a brief description that includes the type and default information for each `get_power` parameter. For a detailed description of the command and all of its parameters, use the man command: `man get_power`
`-include_unit`	Specifies to include the units for the power attributes. *Data_type*: bool, optional
`-insts` *instances_list*	Specifies a list of instance names. The `-nets`, `-pins`, and `-insts` parameters are mutually exclusive. You must specify one, and only one, of the parameters. **Note:** You can use the `get_cells` command to determine the power attributes for a collection of instances. *Data_type*: string, optional

`-nets nets_list`	Specifies a list of net names. The `-nets`, `-pins`, and `-insts` parameters are mutually exclusive. You must specify one, and only one, of the parameters. **Note:** You can use the `get_nets` command to determine the power attributes for a collection of nets. *Data_type*: string, optional
`-out_file filename`	Specifies the output file into which the power related properties are to be written. The `-tcl_list` and `-out_file` parameters are mutually exclusive; you cannot specify them together. *Data_type*: string, optional
`-pg_net {pg_net_name_list | all}`	
	Specifies to query various power-related properties of a specified power net for an instance. This parameter supports the argument all to query properties of all power nets connected to the instance. You must use the `-insts` parameter in conjunction with the `-pg_net` parameter. The `-pg_net` parameter supports the following attributes: `switching_power`, `internal_power`, `leakage_power`, `total_power`, and `dynamic_power`. *Data_type*: string, optional
`-pins pins_list`	Specifies a list of pin names. The `-nets`, `-pins`, and `-insts` parameters are mutually exclusive. You must specify one, and only one, of the parameters. **Note:** You can use the `get_pins` command to determine the power attributes for a collection of pins. *Data_type*: string, optional
`-tcl_list`	Produces the report in Tcl list format instead of a tabular format. The `-tcl_list` and `-out_file` parameters are mutually exclusive; you cannot specify them together. *Data_type*: bool, optional

The following power attributes can be queried for the design objects:

List of Power Attributes - Attribute Description

- `switching_power (net, inst)`: switching power of a net or instance
- `internal_power (inst)`: internal power of an instance
- `dynamic_power (inst)`: switching and internal power of an instance
- `leakage_power (inst)`: leakage power of an instance
- `total_power (inst)`: total power of an instance
- `toggle_rate (net, pin, inst)`: Toggle rate - Number of toggles in the specified VCD/TCF in the duration
- `duty_cycle (net, pin)`: Static probability of the signal to stay high
- `ref_clock (net, pin, inst)`: Reference clock name (Fastest clock for multi clock domains)
- `ref_clock_period (net, pin, inst)`: Reference clock period
- `transition_density (net, pin)`: toggle_rate*ref_clock_period
- `loading_cap (net, pin, inst)`: Loading capacitance
- `voltage (net, pin, inst)`: Operating voltage for a net or pin. For MSMV instance, this attribute can be used to get breakdown of domain specific power.
- `rise_slew (net, pin)`: Rise slew
- `fall_slew (net, pin)`: Fall slew
- `glitch_rate (net, pin)`: Glitch rate - Number of glitches per duration. Applies only to the vector-driven flow.
- `glitch_power (net, pin, inst)`: Glitch power. Applies only to the vector-driven flow.
- `activity_source (net, pin)`: Source of activity annotation. The options are: propagated, user_annotated, and constant.
- `pin_direction (pin)`: [input | output | inout]
- `cell_type (inst)`:Lists cell type as memory, standard cell, IO, sequential, and so on
- `all_attributes (net, pin, inst)`: Lists all related power attributes for the specified design object.

Examples

- The following command returns the specified power attributes for the net clock1:

  ```
  get_power -nets clock1 -attribute {switching_power internal_power toggle_rate}
  ```

  The following is the sample output:

  ```
  clock1 0.04 NA 2
  ```

- The following command returns the specified power attributes for all nets that match the specified pattern `master*`:

  ```
  get_power -nets [get_nets master*] -attribute {switching_power toggle_rate activity_source
  ```

```
duty_cycle} -include_unit
```

The following is the sample output:

```
masterin[1] 0.001mW 0.2 user_annotated 0.5
masterin[2] 0.002mW 0.2 user_annotated 0.5
masterin[3] 0.003mW 0.1 propagated 0.5
masterin[4] 0.005mW 1 constant 1
```

- The following command returns power-related properties of all power nets connected to the instance `clk_buf_1`:

  ```
  get_power -pg_net all -insts clk_buf_1 -attribute {switching_power internal_power leakage_power}
  ```

  The following is the sample output:

  ```
  clk_buf_1 VDD 0.002 0.001 0.000001
  clk_buf_1 VDDL 0.0001 0.0001 0.0000001
  ```

Related Information

- power Category Attributes
- `report_power`
- "Static Power, IRdrop and EM Analysis" in the *Voltus Stylus Common UI User Guide*

"Dynamic Power and IRDrop Analysis" in the *Voltus Stylus Common UI User Guide*

get_property

```
get_property
var_name
property
[-clock clock_name]
[-view view_name]
[-quiet]
```

Retrieves the attribute value for the specified object(s) property.

You can query the following:

- Cell Properties
- Clock Properties
- Design Properties
- Library Properties
- Library Cell Properties
- Library Pin Properties
- Library Pin Properties for Power/Ground Objects
- Library Timing Arc Properties
- Net Properties
- Net Properties for Power/Ground Objects
- Path Group Properties
- Pin Properties
- Pin Properties for Power/Ground Objects
- Port Properties
- Timing Arc Properties
- Timing Path Properties
- Timing Point Properties
- si_victim Properties
- si_attacker Properties
- User-Defined Properties

The `get_property` command supports DRV constraints (`set_min_fanout`, `set_max_fanout`, `set_min_transition`, `set_max_transition`, `set_min_capacitance`, and `set_max_capacitance`) applied on library pins.

The `get_property` command does not work with partial property names. You are now required to specify the full name of the property.

Parameters

`-clock` *clock_name*	Specifies the clock for which to query certain object properties.
property	Specifies the property for which to retrieve the attribute value.
`-quiet`	Suppresses all error and warning messages generated when the `get_property` command is run.
var_name	Specifies a pointer to a collection containing the object. **Note:** If the collection contains more than one object, the `get_property` command returns the property values for each object in the specified collection.
`-view` *view_name*	Specifies the analysis view in which to query the object attribute. You can query properties only for the following objects in an analysis view: library cells, library pins, ports, cells, pins, timing arcs, and nets. You can specify this parameter only when the software is in multi-mode multi-corner timing analysis mode. **Note**: The properties (described in the tables below), which are related to slack, delay, arrival, slew, transition, capacitance, uncertainty and latency, depend on the specified view. When the `-view` parameter is not specified, the maximum (or minimum) value for all the views is returned.

Cell Properties

`area`	Returns the area of the cell. If the cell is hierarchical, also includes the net area. *Return type:* float
`early_cell_check_derate_factor`	Returns the derating factor for early paths specified through the `set_timing_derate` command with the `-early` parameter. *Return type:* float
`early_cell_check_mean_derate_factor`	Returns the derating factor for early paths on the instance – when the `set_timing_derate -mean` parameter is specified. *Return type:* float
`early_clock_check_incremental_derate_factor`	Returns the early clock path instances' incremental derating factor specified through the `set_timing_derate -incremental_adjust` parameter. *Return type:* float

`early_data_check_incremental_derate_factor`	Returns the early data path instance's incremental derating factor specified through the `set_timing_derate -incremental_adjust` parameter. *Return type:* float
`early_cell_check_sigma_derate_factor`	Returns the derating factor for early paths on the instance - when the `set_timing_derate -sigma` parameter is specified. *Return type:* float
`early_clk_cell_derate_factor`	Returns the derating factor for early clock paths specified through the `set_timing_derate` command with the `-early` parameter. *Return type:* float
`early_clk_cell_mean_derate_factor`	Returns the derating factor for early clock paths – when the `set_timing_derate -mean` parameter is specified. *Return type:* float
`early_clk_cell_sigma_derate_factor`	Returns the derating factor for early clock paths – when the `set_timing_derate -sigma` parameter is specified. *Return type:* float
`early_data_cell_derate_factor`	Returns the derating factor for early paths specified through the `set_timing_derate` command with the `-early` parameter. *Return type:* float
`early_data_cell_mean_derate_factor`	Returns the derating factor for early paths – when the `set_timing_derate -mean` parameter is specified. *Return type:* float
`early_data_cell_sigma_derate_factor`	Returns the derating factor for early paths – when the `set_timing_derate -sigma` parameter is specified. *Return type:* float
`early_fall_cell_check_derate_factor`	Returns the early cell check derating factor specified through the `set_timing_derate` command with the `-fall` parameter. *Return type:* float
`early_fall_cell_check_mean_derate_factor`	Returns the early cell check derating factor – when the `set_timing_derate -cell_check` and -mean parameters are specified. *Return type:* float

`early_fall_cell_check_sigma_derate_factor`	Returns the early cell check derating factor – when the `set_timing_derate -cell_check` and `-sigma` parameters are specified. *Return type:* float
`early_fall_clk_cell_derate_factor`	Returns the early clock path derating factor specified through the `set_timing_derate` command with the `-fall` parameter. *Return type:* float
`early_fall_clk_cell_mean_derate_factor`	Returns the early clock path derating factor – when the `set_timing_derate -fall` and `-mean` parameters are specified. *Return type:* float
`early_fall_clk_cell_sigma_derate_factor`	Returns the early clock path derating factor – when the `set_timing_derate -fall` and `-sigma` parameters are specified. *Return type:* float
`early_fall_clock_incremental_derate_factor`	Returns the early clock path instances' incremental derating factor specified through the `set_timing_derate -incremental_adjust` parameter - when the `-fall` parameter is specified. *Return type*: float
`early_fall_data_cell_derate_factor`	Returns the early data cell check derating factor – when the `set_timing_derate -fall` parameter is specified. *Return type:* float
`early_fall_data_cell_mean_derate_factor`	Returns the early data cell check derating factor – when the `set_timing_derate -fall` and `-mean` parameters are specified. *Return type:* float
`early_fall_data_cell_sigma_derate_factor`	Returns the early data cell check derating factor – when the `set_timing_derate -fall` and `-sigma` parameters are specified. *Return type:* float
`early_fall_data_incremental_derate_factor`	Returns the early data path instances' incremental derating factor specified through the `set_timing_derate -incremental_adjust` parameter - when the `-fall` parameter is specified. *Return type*: float

`early_rise_cell_check_derate_factor`	Returns the early cell check derating factor specified through the `set_timing_derate` command with the `-rise` and `-cell_check` parameter. *Return type:* float
`early_rise_cell_check_mean_derate_factor`	Returns the early cell check derating factor – when the `set_timing_derate -rise` and `-mean` parameters are specified. *Return type:* float
`early_rise_cell_check_sigma_derate_factor`	Returns the early cell check derating factor – when the `set_timing_derate -rise` and `-sigma` parameters are specified. *Return type:* float
`early_rise_clk_cell_derate_factor`	Returns the early clock cell check derating factor specified through the `set_timing_derate` command with the `-rise` parameter. *Return type:* float
`early_rise_clk_cell_mean_derate_factor`	Returns the early clock cell check derating factor – when the `set_timing_derate -rise` and `-mean` parameters are specified. *Return type:* float
`early_rise_clk_cell_sigma_derate_factor`	Returns the early clock cell check derating factor – when the `set_timing_derate -rise` and `-sigma` parameters are specified. *Return type:* float
`early_rise_clock_incremental_derate_factor`	Returns the early clock path instances' incremental derating factor specified through the `set_timing_derate -incremental_adjust` parameter - when the `-rise` parameter is specified. *Return type*: float
`early_rise_data_cell_derate_factor`	Returns the early data cell check derating factor specified through the `set_timing_derate` command with the `-rise` parameter. *Return type:* float
`early_rise_data_cell_mean_derate_factor`	Returns the early data cell check derating factor – when the `set_timing_derate -rise` and `-mean` parameters are specified. *Return type:* float
`early_rise_data_cell_sigma_derate_factor`	Returns the early data cell check derating factor – when the `set_timing_derate -rise` and `-sigma` parameters are specified. *Return type:* float

`early_rise_data_incremental_derate_factor`	Returns the early data path instances' incremental derating factor specified through the `set_timing_derate -incremental_adjust` parameter - when the `-rise` parameter is specified. *Return type*: float
`hierarchical_name`	Returns the complete hierarchical name of the cell. *Return type:* string
`instance_model`	Reports the model type of an instance. The Instance model type can be ecsmn, ccsn, cdb, udn, or nldm. In case a noise model does not exist, the software returns 'NA'. For example, `get_property [get_cells inst1/buf22 ] instance_model` *Return type:* string
`is_black_box`	Returns a value of `true` if the cell is a black box cell. A cell is considered as a black box if the "function" definition of the cell is missing from the library. *Return type:* boolean
`is_buffer`	Returns a value of `true` if the associated library cell is recognized as a buffer cell. For example, `get_property [get_cells mmu/IB1] is_buffer` *Return type:* boolean
`is_clock_gating_check`	Returns a value of true if the cell is gated. *Return type:* boolean
`is_combinational`	Returns a value of `true` if the cell is a combinational cell (not a sequential cell). *Return type:* boolean
`is_disable_timing`	Returns a value of `true` if the cell's timing has been disabled using the `set_disable_timing` command. *Return type:* boolean
`is_dont_touch`	Returns a value of `true` if the cell definition includes `dont_touch:true`. *Return type:* boolean

`is_fall_edge_triggered`	Returns a value of `true` if the cell is triggered by the falling edge of the clock. *Return type:* boolean
`is_hierarchical`	Returns a value of `false` if the cell is a leaf or library cell. *Return type:* boolean
`is_integrated_clock_gating_cell`	Returns a value of `true` if the cell is an integrated clock gating cell. *Return type:* boolean
`is_interface_timing`	Returns a value of true if a library cell has interface timing specified for that cell. *Return Type*: boolean
`is_memory_cell`	Returns a value of `true` if the cell is a memory cell. *Return type:* boolean
`is_negative_level_sensitive`	Returns a value of `true` if the cell is used as a negative level-sensitive sequential element, such as a negative level-sensitive latch. *Return type:* boolean
`is_pad_cell`	Returns a value of `true` if the Liberty library `is_pad_cell` attribute is set to `true` for this pad cell. *Return type:* boolean
`is_positive_level_sensitive`	Returns a value of `true` if the cell is used as a positive level-sensitive sequential element, such as a positive level-sensitive latch. *Return type:* boolean
`is_rise_edge_triggered`	Returns a value of `true` if the cell is triggered by the rising edge of the clock. *Return type:* boolean
`is_sequential`	Returns a value of `true` if the cell is a latch or flip-flop, or if the cell has sequential timing checks. *Return type:* boolean
`is_tristate`	Returns a value of `true` if the cell definition includes `three_state:true`. *Return type:* boolean

`late_cell_check_derate_factor`	Returns the derating factor for late paths specified through the `set_timing_derate` command with the `-late` parameter. *Return type:* float
`late_cell_check_mean_derate_factor`	Returns the derating factor for late paths – when the `set_timing_derate -late` and `-mean` parameters are specified. *Return type:* float
`late_cell_check_sigma_derate_factor`	Returns the derating factor for late paths – when the `set_timing_derate -late` and `-sigma` parameters are specified. *Return type:* float
`late_clk_cell_derate_factor`	Returns the derating factor for late clock paths specified through the `set_timing_derate` command with the `-late` parameter. *Return type:* float
`late_clk_cell_mean_derate_factor`	Returns the derating factor for late clock paths – when the `set_timing_derate -late` and `-mean` parameters are specified. *Return type:* float
`late_clk_cell_sigma_derate_factor`	Returns the derating factor for late clock paths – when the `set_timing_derate -late` and `-sigma` parameters are specified. *Return type:* float
`late_clock_check_incremental_derate_factor`	Returns the late clock path instance's incremental derating factor specified through the `set_timing_derate -incremental_adjust` parameter. *Return type*: float
`late_data_cell_derate_factor`	Returns the derating factor for late data paths specified through the `set_timing_derate` command with the `-late` parameter. *Return type:* float
`late_data_cell_sigma_derate_factor`	Returns the derating factor for late data paths – when the `set_timing_derate -late` and `-sigma` parameters are specified. *Return type:* float
`late_data_check_incremental_derate_factor`	Returns the late data path instances' incremental derating factor specified through the `set_timing_derate -incremental_adjust` parameter. *Return type*: float

`late_fall_cell_check_derate_factor`	Returns the late cell check derating factor specified through the `set_timing_derate` command with the `-fall` parameter. *Return type:* float
`late_fall_cell_check_mean_derate_factor`	Returns the late cell check derating factor – when the `set_timing_derate -fall` and `-mean` parameters are specified. *Return type:* float
`late_fall_cell_check_sigma_derate_factor`	Returns the late cell check derating factor – when the `set_timing_derate -fall` and `-sigma` parameters are specified. *Return type:* float
`late_fall_clk_cell_derate_factor`	Returns the late clock cell check derating factor specified through the `set_timing_derate` command with the `-fall` parameter. *Return type:* float
`late_fall_clk_cell_mean_derate_factor`	Returns the late clock cell check derating factor – when the `set_timing_derate -fall` and `-mean` parameters are specified. *Return type:* float
`late_fall_clk_cell_sigma_derate_factor`	Returns the late clock cell check derating factor – when the `set_timing_derate -fall` and `-sigma` parameters are specified. *Return type:* float
`late_fall_clock_incremental_derate_factor`	Returns the late clock path instances' incremental derating factor specified through the `set_timing_derate -incremental_adjust` parameter - when the `-fall` parameter is specified. *Return type*: float
`late_fall_data_cell_derate_factor`	Returns the late data cell check derating factor specified through the `set_timing_derate` command with the `-fall` parameter. *Return type:* float
`late_fall_data_cell_mean_derate_factor`	Returns the late data cell check derating factor – when the `set_timing_derate -fall` and `-mean` parameters are specified. *Return type:* float
`late_fall_data_cell_sigma_derate_factor`	Returns the late data cell check derating factor – when the `set_timing_derate -fall` and `-sigma` parameters are specified. *Return type:* float

`late_fall_data_incremental_derate_factor`	Returns the late data path instances' incremental derating factor specified through the `set_timing_derate -incremental_adjust` parameter - when the `-fall` parameter is specified. Return type: float
`late_rise_cell_check_derate_factor`	Returns the late cell check derating factor specified through the `set_timing_derate` command with the `-rise` parameter. *Return type:* float
`late_rise_cell_check_mean_derate_factor`	Returns the late cell check derating factor – when the `set_timing_derate -rise` and `-mean` parameters are specified. *Return type:* float
`late_rise_cell_check_sigma_derate_factor`	Returns the late cell check derating factor – when the `set_timing_derate -fall` and `-sigma` parameters are specified. *Return type:* float
`late_rise_clk_cell_derate_factor`	Returns the late clock cell check derating factor specified through the `set_timing_derate` command with the `-rise` parameter. *Return type:* float
`late_rise_clk_cell_mean_derate_factor`	Returns the late clock cell check derating factor – when the `set_timing_derate -rise` and `-mean` parameters are specified. *Return type:* float
`late_rise_clk_cell_sigma_derate_factor`	Returns the late clock cell check derating factor – when the `set_timing_derate -fall` and `-sigma` parameters are specified. *Return type:* float
`late_rise_clock_incremental_derate_factor`	Returns the late clock path instances' incremental derating factor specified through the `set_timing_derate -incremental_adjust` parameter - when the `-rise` parameter is specified. *Return type*: float
`late_rise_data_cell_derate_factor`	Returns the late data cell check derating factor specified through the `set_timing_derate` command with the `-rise` parameter. *Return type:* float
`late_rise_data_cell_mean_derate_factor`	Returns the late data cell check derating factor – when the `set_timing_derate -rise` and `-mean` parameters are specified. *Return type:* float

`late_rise_data_cell_sigma_derate_factor`	Returns the late data cell check derating factor – when the `set_timing_derate -rise` and `-sigma` parameters are specified. *Return type:* float
`late_rise_data_incremental_derate_factor`	Returns the late data path instances' incremental derating factor specified through the `set_timing_derate -incremental_adjust` parameter - when the `-rise` parameter is specified. *Return type*: float
`lib_cell`	Returns a collection of all the library cell objects associated with the specified library pins/cells. For example, `get_property [get_cells AAA] lib_cell -view VIEW1` *Return type*: collection
`name`	Returns the leaf-level name of the cell. *Return type:* string
`object_type`	Returns the object type `cell`. *Return type:* string
`pin_count`	Returns the total number of signal pins. *Return type:* integer
`ref_lib_cell_name`	Returns the name of reference module of the given cell. The reference module of the cell can be a Verilog module or a leaf library cell. *Return type:* string
`threshold_voltage_group`	Returns name of a group/category – that is specified based on the cell's threshold voltage (high or low) characteristics. For example, `get_property [get_lib_cells AND2_X1] threshold_voltage_group` *Return type:* string
`timing_model_type`	Returns the model type for a given cell or instance. The supported values are `abstracted`, `extracted`, and `qtm`. *Return type:* string
`x_coordinate_max`	Returns the maximum x coordinate of the cell. *Return type:* float
`x_coordinate_min`	Returns the minimum x coordinate of the cell. *Return type:* float

`y_coordinate_max`	Returns the maximum y coordinate of the cell. *Return type:* float
`y_coordinate_min`	Returns the minimum y coordinate of the cell. *Return type:* float

Cell Properties for Power/Ground Objects	
`is_always_on`	Returns a value of `true` if a cell instance is an always-on cell. *Return type*: boolean For example, `get_property [get_cells mmu/U1] is_always_on`
`is_inverter`	Returns a value of `true` if a cell instance is an inverter cell. *Return type*: boolean For example, `get_property [get_cells mmu/INV1] is_inverter`
`is_isolation_cell`	Returns a value of `true` if a cell instance is an isolation cell. *Return type*: boolean For example, `get_property [get_cells mmu/U1] is_isolation_cell`
`is_level_shifter`	Returns a value of `true` if a cell instance is a level-shifter cell. *Return type*: boolean For example, `get_property [get_cells mmu/U1] is_level_shifter`
`is_power_switch`	Returns a value of `true` if a cell instance is a power switch cell. Return type: boolean For example, `get_property [get_cells mmu/U1] is_power_switch`
`is_retention`	Returns a value of `true` if a cell instance is a retention cell. *Return type*: boolean For example, `get_property [get_cells mmu/U1] is_retention`
`level_shifter_type`	Returns the type of level-shifter (e.g., LH, HL, or HL_LH) that a cell instance is. *Return type*: string For example, `get_property [get_cells mmu/U1] level_shifter_type`
`pg_pins`	Returns a collection of the PG pins that are present in a cell instance. *Return type*: collection For example, `get_property [get_cells mmu/U1] pg_pins`

`power_switch_type`	Returns the type of power switch (e.g., coarse-grain, fine-grain) that a cell instance is. *Return type*: string For example, `get_property [get_cells mmu/U1] power_switch_type`
`std_cell_main_rail_name`	Returns the name of the main power rail driving the power pins of a cell instance that is the `pg_pin`, which has the `std_cell_main_rail` library attribute set as `true` for the instance cell. *Return type*: string For example, `get_property [get_cells mmu/U1] std_cell_main_rail_name`

Clock Properties

`clock_network_pins`	Returns a collection of all pins on the clock network for the clock. *Return type:* collection
`delay_max_fall`	Returns the maximum falling delay value for the clock. *Return type:* float
`delay_max_rise`	Returns the maximum rising delay value for the clock. *Return type:* float
`delay_min_fall`	Returns the minimum falling delay value for the clock. *Return type:* float
`delay_min_rise`	Returns the minimum rising delay value for the clock. *Return type:* float
`generated_clocks_extended`	Returns generated clocks derived from the regular clock. *Return type:* collection
`hierarchical_name`	Returns the complete hierarchical name of the clock. *Return type:* string
`ideal_transition_max_fall`	Returns the ideal clock transition asserted on clock waveforms specified with the `set_clock_transition` command using the `-fall` and `-max` parameters. *Return type:* float
`ideal_transition_max_rise`	Returns the ideal clock transition asserted on clock waveforms specified with the `set_clock_transition` command using the `-rise` and `-max` parameters. *Return type:* float

`ideal_transition_min_fall`	Returns the ideal clock transition asserted on clock waveforms specified with `set_clock_transition` command using the `-fall` and `-min` parameters. *Return type:* float
`ideal_transition_min_rise`	Returns the ideal clock transition asserted on clock waveforms specified with `set_clock_transition` command using the `-rise` and `-min` parameters. *Return type:* float
`is_active`	Returns a value of `true` if the clock specified is active in the current view. *Return type:* boolean
`is_generated`	Returns a value of `true` if the clock is a generated clock. *Return type:* boolean
`is_propagated_clock`	Returns a value of `true` if the clock is in propagated mode. *Return type:* boolean
`latency_fall_max`	Returns the maximum fall latency for the clock. *Return type:* float
`latency_fall_min`	Returns the minimum fall latency for the clock. *Return type:* float
`latency_rise_max`	Returns the maximum rise latency for the clock. *Return type:* float
`latency_rise_min`	Returns the minimum rise latency for the clock. *Return type:* float
`max_capacitance_clock_path_fall`	Returns fall max_capacitance value on clock paths. *Return type:* float
`max_capacitance_clock_path_rise`	Returns rise max_capacitance value on clock paths. *Return type:* float
`max_capacitance_data_path_fall`	Returns fall max_capacitance value on data paths. *Return type:* float
`max_capacitance_data_path_rise`	Returns rise max_capacitance value on data paths. *Return type:* float
`max_transition_clock_path_fall`	Returns fall max_transition value on clock paths. *Return type:* float

`max_transition_clock_path_rise`	Returns rise max_transition value on clock paths. *Return type:* float
`max_transition_data_path_fall`	Returns fall max_transition value on data paths. *Return type:* float
`max_transition_data_path_rise`	Returns rise max_transition value on data paths. *Return type:* float
`name`	Returns the name of the clock. *Return type:* string
`object_type`	Returns the object type `clock`. *Return type:* string
`period`	Returns the clock period. *Return type:* float
`source_latency_early_fall_max`	Returns the early source latency for the falling transition at the max corner. *Return type:* float
`source_latency_early_fall_min`	Returns the early source latency for the falling transition at the min corner. *Return type:* float
`source_latency_early_rise_max`	Returns the early source latency for the rising transition at the max corner. *Return type:* float
`source_latency_early_rise_min`	Returns the early source latency for the rising transition at the min corner. *Return type:* float
`source_latency_late_fall_max`	Returns the late source latency for the falling transition at the max corner. *Return type:* float
`source_latency_late_fall_min`	Returns the late source latency for the falling transition at the min corner. *Return type:* float
`source_latency_late_rise_max`	Returns the late source latency for the rising transition at the max corner. *Return type:* float
`source_latency_late_rise_min`	Returns the late source latency for the rising transition at the min corner. *Return type:* float
`sources`	Returns a collection of the clock's source pins or ports. *Return type:* collection

`view_name`	In multi-mode multi-corner mode, returns the analysis view name associated with a given clock. *Return type*: string
`waveform`	Returns the rising and falling edge times of the waveform. *Return type:* string

Design Properties

`analysis_type`	Returns the analysis mode: `single`, `bc_wc`, or `on_chip_variation`. *Return type:* string
`capacitance_unit_in_farad`	Returns the capacitance unit specified in the library. *Return type:* float
`chip_size`	Returns the chip size. The chip size can be specified with the `timing_aocv_chip_size` attribute. If this attribute is not set, then the software will calculate the chip size from the design node locations. This setting takes precedence over the software setting. *Return type:* float
`current_unit_in_amp`	Returns the current unit specified in the library. *Return type:* float
`hierarchical_name`	Returns the complete hierarchical name of the design. *Return type:* string
`is_dont_touch`	Returns `true` if the design is specified through the `set_dont_touch` command. *Return type:* boolean
`is_max_transition_asserted`	Returns a value of `true` if a maximum transition value (set using the `set_max_transition` command) is defined in the design. *Return type:* boolean
`is_min_transition_asserted`	Returns a value of `true` if a minimum transition value (set using the `set_min_transition` command) is defined in the design. *Return type:* boolean
`max_capacitance`	Returns the maximum capacitance value for the design. *Return type:* float

`max_fanout`	Returns the maximum fanout value for the design. *Return type:* float
`max_transition`	Returns the maximum transition design rule limit for the design. *Return type:* float
`min_capacitance`	Returns the minimum capacitance value for the design. *Return type:* float
`object_type`	Returns the object type `design`. *Return type: string*
`operating_condition_max`	Returns the name of the maximum operating condition specified for the design. *Return type:* string
`operating_condition_min`	Returns the name of the minimum operating condition specified for the design. *Return type:* string
`process_max`	Returns the process value of the maximum operating condition specified for the design. *Return type:* float
`process_min`	Returns the process value of the minimum operating condition specified for the design. *Return type:* float
`resistance_unit_in_ohm`	Returns the resistance value from the library. *Return type:* float
`slew_threshold_percent_fall_high`	Returns the upper slew trip point for falling waveforms, specified as a percent. *Return type:* float
`slew_threshold_percent_fall_low`	Returns the lower slew trip point for falling waveforms, specified as a percent. *Return type:* float
`slew_threshold_percent_rise_high`	Returns the upper slew trip point for rising waveforms, specified as a percent. *Return type:* float
`slew_threshold_percent_rise_low`	Returns the lower slew trip point for rising waveforms, specified as a percent. *Return type:* float
`temperature_max`	Returns the temperature value of the maximum operating condition for the design. *Return type:* float

`temperature_min`	Returns the temperature value of the minimum operating condition for the design. *Return type:* float
`time_unit_in_second`	Returns the time units of the design, in seconds. that is, time unit of the design is 1.000000ns, the property returns 1.000000e-09. *Return type:* float
`tree_type_max`	Returns the value of the `tree_type` attribute for the specified maximum operating condition. *Return type:* string
`tree_type_min`	Returns the value of the `tree_type` attribute for the specified minimum operating condition. *Return type:* string
`voltage_max`	Returns the voltage value of the maximum operating condition for the design. *Return type:* float
`voltage_min`	Returns the voltage value of the minimum operating condition for the design. *Return type:* float
`voltage_unit_in_volt`	Returns the voltage units specified in the library for the design. *Return type:* float

Library Properties

`default_max_transition`	Returns the default maximum transition time limit specified for a given library. *Return type:* float
`hierarchical_name`	Returns the hierarchical path name for the library. *Return type:* string
`input_threshold_pct_fall`	Returns the value of threshold point on an input pin signal falling from 1 to 0, which is used in modelling the delay of a signal transmitting from an input pin to an output pin. *Return type:* integer
`input_threshold_pct_rise`	Returns the value of threshold point on an input pin signal rising from 0 to 1, which is used in modelling the delay of a signal transmitting from an input pin to an output pin. *Return type:* integer

`lib_file_path`	Reports full on-disk path of a library. In case of a library database (LDB) file, the property will return the full on-disk path of the original library from which LDB was compiled. *Return type:* string
`lib_version`	Returns the software version in which the library file (LDB) was generated. *Return type:* integer
`nominal_process`	Returns a value of the process either modeled as nominal process or as the process in default operating condition in the library. *Return Type:* float For example, `get_property [get_libs sc_cmoslp] nominal_process`
`nominal_temperature`	Returns a value of the temperature either modeled as nominal temperature or as the temperature in default operating condition in the library. *Return Type:* float For example, `get_property [get_libs sc_cmoslp] nominal_temperature`
`nominal_voltage`	Returns a value of the voltage either modeled as nominal voltage or as the voltage in default operating condition in the library. *Return Type:* float For example, `get_property [get_libs sc_cmoslp] nominal_voltage`
`object_type`	Returns the object type `library`. *Return type:* string
`output_threshold_pct_fall`	Returns the value of threshold point on an output signal falling from 1 to 0, which is used in modelling the delay of a signal transmitting from an input pin to an output pin. *Return type:* integer
`output_threshold_pct_rise`	Returns the value of threshold point on an output signal rising from 0 to 1, which is used in the modelling of a signal transmitting from an input pin to an output pin. *Return type:* integer
`slew_derate_from_library`	Specifies how the transition times found in the library need to be derated to match the transition times between the characterization trip points. *Return type:* float
`slew_lower_threshold_pct_fall`	Returns the value of lower threshold point used in modeling the delay of a pin falling from 1 to 0. *Return type:* float

`slew_lower_threshold_pct_rise`	Returns the value of lower threshold point used to model the delay of a pin rising from 0 to 1. *Return type:* float
`slew_upper_threshold_pct_fall`	Returns the value of upper threshold point used to model the delay of a pin falling from 1 to 0. *Return type:* float
`slew_upper_threshold_pct_rise`	Returns the value of upper threshold point used to model the delay of a pin rising from 0 to 1. *Return type:* float
`source_file_name`	Returns the full path of all library files that contain the specified library name. The use model is given below: `get_property [get_libs libname] source_file_name [-view viewName]` If the `-view` parameter is specified, the property returns the path of the library files in the given analysis view. If this parameter is not specified, the command returns the path of all library files in all the analysis views. *Return Type:* string
Library Properties for Power/Ground Objects	

`power_rails`	Returns the name of the power rails along with their corresponding voltages as defined in the `voltage_map` attribute in the library.
	Return Type: string
	For example,
	`get_property [get_libs sc_cmoslp] power_rails`
	`Output: Is a list of lists, each list showing the power rail name along with its associated voltage.`
	`{ {VDDG 1.045} {VDD 1.045} {VSS 0} }`

Library Cell Properties

`area`	Returns the area of the library cell. *Return Type:* float

`early_clock_check_incremental_derate_factor`	Returns the early clock path library cell check incremental derating factor specified through the `set_timing_derate -incremental_adjust` parameter. *Return Type:* float
`early_data_check_incremental_derate_factor`	Returns the early data path library cell check incremental derating factor specified through the `set_timing_derate -incremental_adjust` parameter. *Return Type*: float
`early_fall_clock_incremental_derate_factor`	Returns the early clock path library cell check incremental derating factor specified through the `set_timing_derate -incremental_adjust` parameter - when the `-fall` parameter is specified. *Return Type*: float
`early_fall_data_incremental_derate_factor`	Returns the early data path library cell check incremental derating factor specified through the `set_timing_derate -incremental_adjust` parameter - when the `-fall` parameter is specified. *Return Type*: float
`early_rise_clock_incremental_derate_factor`	Returns the early clock path library cell check incremental derating factor specified through the `set_timing_derate -incremental_adjust` parameter - when the `-rise` parameter is specified. *Return Type*: float
`early_rise_data_incremental_derate_factor`	Returns the early data path library cell check incremental derating factor specified through the `set_timing_derate -incremental_adjust` parameter - when the `-rise` parameter is specified. *Return Type*: float
`has_spatial`	Returns a value of `true` if a spatial derate exists for a library cell. *Return type:* boolean
`hierarchical_name`	Returns the complete hierarchical name of the library cell (the library name followed by the library cell name). *Return Type:* string
`is_black_box`	Returns a value of `true` if the cell is a black box cell. A cell is considered as a black box if the "function" definition of the cell is missing from the library. *Return type:* boolean

`is_combinational`	Returns a value of `true` if the library cell is a combinational cell (not a sequential cell). *Return type:* boolean
`is_dont_touch`	Returns a value of `true` if the library cell definition includes `dont_touch:true`. *Return Type:* boolean
`is_dont_use`	Returns a value of `true` if the library cell description includes `dont_use:true`. *Return Type:* boolean
`is_fall_edge_triggered`	Returns a value of `true` if the library cell is triggered by the falling edge of the clock. *Return Type:* boolean
`is_interface_timing`	Returns a value of `true` if a library cell has interface timing specified for that cell.

`is_memory_cell`	Returns a value of `true` if the cell is a memory cell.
	Return type: boolean
	Note: The library cells that have a memory group defined in the dotlib are considered as memory cells. For example:
	`cell ( ) {`
	`:`
	`memory() {`
	`type : ram ;`
	`address_width : 10 ;`
	`word_width : 7 ;`
	`}`
	`}`

`is_negative_level_sensitive`	Returns a value of `true` if the library cell is used as a negative level-sensitive latch. *Return type:* boolean
`is_pad_cell`	Returns a value of `true` if the Liberty library `is_pad_cell` attribute is set to `true` for this pad cell. *Return type:* boolean

`is_pll_cell`	Returns a value of `true` if the Liberty library `is_pll_cell` attribute is set to `true` for this library cell. *Return type:* boolean
`is_positive_level_sensitive`	Returns a value of `true` if the library cell is used as a positive level-sensitive latch. *Return type:* boolean
`is_rise_edge_triggered`	Returns a value of `true` if the library cell is triggered by the rising edge of the clock. *Return Type:* boolean
`is_sequential`	Returns a value of `true` if the library cell is a latch or flip-flop, or if the cell has sequential timing checks. *Return type:* boolean
`is_tristate`	Returns a value of `true` if the library cell definition includes `three_state:true`. *Return type:* boolean
`late_clock_check_incremental_derate_factor`	Returns the late clock path library cell check incremental derating factor specified through the `set_timing_derate -incremental_adjust` parameter. *Return Type*: float
`late_data_check_incremental_derate_factor`	Returns the late data path library cell check incremental derating factor specified through the `set_timing_derate -incremental_adjust` parameter. *Return Type*: float
`late_fall_clock_incremental_derate_factor`	Returns the late clock path library cell check incremental derating factor specified through the `set_timing_derate -incremental_adjust` parameter - when the -fall parameter is specified. *Return Type*: float
`late_fall_data_incremental_derate_factor`	Returns the late data path library cell check incremental derating factor specified through the `set_timing_derate -incremental_adjust` parameter - when the `-fall` parameter is specified. *Return Type*: float
`late_fall_clock_period_derate_factor`	Returns the derate factor for the clock period when the `set_timing_derate -late` and `-fall` parameters are specified. *Return type:* float

`late_fall_clock_period_mean_derate_factor`	Returns the late/fall derate factor for the clock period when the `set_timing_derate -mean` parameter is specified. *Return type:* float
`late_fall_clock_period_sigma_derate_factor`	Returns the late/fall derate factor for the clock period when the `set_timing_derate -sigma` parameter is specified. *Return type:* float
`late_fall_pulse_width_derate_factor`	Returns the late/fall derate factor for the pulse width when the `set_timing_derate` -late and -fall parameters are specified. *Return type:* float
`late_fall_pulse_width_mean_derate_factor`	Returns the late/fall derate factor for the pulse width when the `set_timing_derate` -mean parameter is specified. *Return type:* float
`late_fall_pulse_width_sigma_derate_factor`	Returns the late/fall derate factor for the pulse width when the `set_timing_derate` -sigma parameter is specified. *Return type:* float
`late_rise_clock_incremental_derate_factor`	Returns the late clock path library cell check incremental derating factor specified through the `set_timing_derate -incremental_adjust` parameter - when the `-rise` parameter is specified. *Return Type*: float
`late_rise_data_incremental_derate_factor`	Returns the late data path library cell check incremental derating factor specified through the `set_timing_derate -incremental_adjust` parameter - when the `-rise` parameter is specified. *Return Type*: float
`late_rise_clock_period_derate_factor`	Returns the derate factor for the clock period when the `set_timing_derate` -late and -rise parameters are specified. *Return type:* float
`late_rise_clock_period_mean_derate_factor`	Returns the late/rise derate factor for the clock period when the `set_timing_derate -mean` parameter is specified. *Return type:* float
`late_rise_clock_period_sigma_derate_factor`	Returns the late/rise derate factor for the clock period when the `set_timing_derate -sigma` parameter is specified. *Return type:* float

`late_rise_pulse_width_derate_factor`	Returns the derate factor for the pulse width check when the `set_timing_derate -late` and `-rise` parameters are specified. *Return type:* float
`late_rise_pulse_width_mean_derate_factor`	Returns the late/rise derate factor for the pulse width when the `set_timing_derate -mean` parameter is specified. *Return type:* float
`late_rise_pulse_width_sigma_derate_factor`	Returns the late/rise derate factor for the pulse width when the `set_timing_derate -sigma` parameter is specified. *Return type:* float
`name`	Returns the leaf-level name of the library cell. *Return Type:* string
`object_type`	Returns the object type `lib_cell`. *Return Type:* string
`receiver_capacitance_type`	Returns the value of input capacitance group being used for delay calculations – C1C2 (2-piece model) or C1CN (n-piece model). *Return Type:* string
`retention_cell`	Returns the value of the retention_cell attribute specified in the liberty. *Return type*: string
`timing_model_type`	Returns the model type for a given cell or instance. The supported values are `abstracted`, `extracted`, and `qtm`. *Return type*: string
Library Cell Properties for Power/Ground Objects	
`ground_pins`	Returns a collection of the ground pins of a library cell. *Return type*: collection For example, `get_property [get_lib_cells sc_cmoslp/AO22X] ground_pins`
`is_always_on`	Returns a value of `true` if the `always_on` attribute of a library cell is set to `true` indicating that the library cell is an always-on cell. *Return type*: boolean For example, `get_property [get_lib_cells sc_cmoslp/AON22_X4M] is_always_on`

`is_isolation_cell`	Returns a value of `true` if the library cell is an isolation cell. *Return type*: boolean For example, `get_property [get_lib_cells sc_cmoslp/ISO22_X4M] is_isolation_cell`
`is_power_switch`	Returns a value of `true` when the `switch_cell_type` attribute is defined for a library cell indicating that the library cell is a power-switch cell. *Return type*: boolean For example, `get_property [get_lib_cells sc_cmoslp/AO22X] is_power_switch`
`is_retention`	Returns a value of `true` when the `retention_cell` attribute is defined for a library cell indicating that the library cell is a retention cell. *Return type*: boolean For example, `get_property [get_lib_cells sc_cmoslp/RET22_X4M] is_retention`
`level_shifter_type`	Returns the type of level-shifter (e.g., LH, HL, or HL_LH) as defined by the `level_shifter_type` attribute of a library cell. *Return type*: string For example, `get_property [get_lib_cells sc_cmoslp/A2LVLD_X4M] level_shifter_type`
`lib_pg_pins`	Returns a collection of all PG pins (both power and ground) defined in the library of a library cell. *Return type*: collection For example, `get_property [get_lib_cells sc_cmoslp/AO22X] lib_pg_pins`
`power_pins`	Returns a collection of the power pins of a library cell. *Return type*: collection For example, `get_property [get_lib_cells sc_cmoslp/AO22X] power_pins`
`power_switch_type`	Returns the type of power-switch (e.g., coarse-grain, fine-grain) as defined by the `switch_cell_type` attribute of library cells. *Return type*: string For example, `get_property [get_lib_cells sc_cmoslp/AO22X] power_switch_type`

`std_cell_main_rail_name`	Returns the name of a power rail that has the `std_cell_main_rail` library attribute defined as `true` for a library cell. *Return type*: string For example, `get_property [get_lib_cells sc_cmoslp/AO22X] std_cell_main_rail_name`

Library Pin Properties

`capacitance`	Returns the capacitance for the library pin. *Return type:* float
`capacitance_max_fall`	Returns the maximum value of the falling capacitance range. *Return type:* float
`capacitance_max_rise`	Returns the maximum value of the rising capacitance range. *Return type:* float
`capacitance_min_fall`	Returns the minimum value of the falling capacitance range. *Return type:* float
`capacitance_min_rise`	Returns the minimum value of the rising capacitance range. *Return type:* float
`direction`	Returns the direction of the library pin: `in`, `out`, or `inout`. *Return type:* string
`fanout_load`	Returns the fanout load value of the library pin. *Return type:* float
`hierarchical_name`	Returns the hierarchical name of the library pin. *Return type:* string
`is_async`	Returns a value of `true` if the library pin is an asynchronous preset pin, or an asynchronous clear pin. *Return type:* boolean
`is_clear`	Returns a value of `true` if the pin is an asynchronous clear pin. *Return type:* boolean

`is_clock`	Returns a value of `true` if the library pin definition includes `clock:true`. *Return type:* boolean
`is_data`	Returns a value of `true` if the library pin with defined sequential arcs is the data pin of a flip-flop. *Return type:* boolean
`is_fall_edge_triggered_clock`	Returns a value of `true` if the library pin is the clock pin of a falling edge triggered device. *Return type:* boolean
`is_fall_edge_triggered_data`	Returns a value of `true` if the library pin is the data pin of a falling edge triggered device. *Return type:* boolean
`is_negative_level_sensitive_clock`	Returns a value of `true` if the library pin is an enable pin of an active low level-sensitive device. *Return type:* boolean
`is_negative_level_sensitive_data`	Returns a value of `true` if the pin is a data pin of an active low level-sensitive device. *Return type:* boolean
`is_pll_feedback_pin`	Returns a value of `true` if the Liberty library `is_pll_feedback_pin` attribute is set to `true` for this library pin. *Return type:* boolean
`is_pll_output_pin`	Returns a value of `true` if the Liberty library `is_pll_output_pin` attribute is set to `true` for this library pin. *Return type:* boolean
`is_pll_reference_pin`	Returns a value of `true` if the Liberty library `is_pll_reference_pin` attribute is set to `true` for this library pin. *Return type:* boolean
`is_positive_level_sensitive_clock`	Returns a value of `true` if the library pin is an enable pin of an active high level-sensitive device. *Return type:* boolean
`is_positive_level_sensitive_data`	Returns a value of `true` if the pin is a data pin of an active high level-sensitive device. *Return type:* boolean

`is_preset`	Returns a value of `true` if the pin is a preset pin. *Return type:* boolean
`is_rise_edge_triggered_clock`	Returns a value of `true` if the library pin is the clock pin of a rising edge triggered device. *Return type:* boolean
`is_rise_edge_triggered_data`	Returns a value of `true` if the library pin is the data pin of a rising edge triggered device. *Return type:* boolean
`is_tristate`	Returns a value of `true` if the library pin definition includes `three_state:true`. *Return type:* boolean
`max_capacitance`	Returns the maximum capacitance limit for the library pin. *Return type:* float
`max_fanout`	Returns the maximum fanout value for the library pin. *Return type:* float
`max_transition`	Returns the maximum transition time limit specified for a given library pin. If the limit is not specified for a library pin, then the limit specified at the corresponding library level will be used. *Return type:* float
`min_transition`	Returns the minimum transition time limit specified for a given library pin. If the limit is not specified for a library pin, then the limit specified at the corresponding library level will be used. *Return type:* float
`min_capacitance`	Returns the minimum capacitance limit for the library pin. *Return type:* float
`min_fanout`	Returns the minimum fanout value for the library pin. *Return type:* float
`name`	Returns the leaf-level name of the library cell pin. *Return type:* string
`object_type`	Returns the object type; in this case, `lib_pin`. *Return type:* string

`related_ground_pin_name`	Returns related ground pin names for the corresponding library pins. *Return type:* string For example, `get_property [get_lib_pins sc_cmoslp/AO22X/I1] related_ground_pin_name`
`related_power_pin_name`	Returns related power pin names for the corresponding library pins. *Return type:* string
`slew_threshold_percent_fall_high`	Returns the upper slew trip point for falling waveforms, specified as a percent. *Return type:* float
`slew_threshold_percent_fall_low`	Returns the lower slew trip point for falling waveforms, specified as a percent. *Return type:* float
`slew_threshold_percent_rise_high`	Returns the upper slew trip point for rising waveforms, specified as a percent. *Return type:* float
`slew_threshold_percent_rise_low`	Returns the lower slew trip point for rising waveforms, specified as a percent. *Return type:* float
Library Signal Pin Properties for Power/Ground Objects	
`input_signal_level_voltage`	Returns the voltage value corresponding to the power rail name/signal which is defined for the library-cell input pin. This voltage value is the full rail-to-rail voltage value that is associated with the relative power rail name through the `voltage_map` attribute defined in the library. *Return type:* float For example, `get_property [get_lib_pins sc_cmoslp/AO22X/I1] input_signal_level_voltage`
`is_always_on`	Returns a value of `true` if the `always_on` attribute of a library signal pin is set to `true` indicating that the library cell signal pin is an always-on pin. *Return type:* boolean For example, `get_property [get_lib_pins sc_cmoslp/AO22X/I1] is_always_on`

`is_isolated`	Returns a value of `true` if the `is_isolated` attribute of a library signal pin is set to `true` indicating that the library cell signal pin is an internally isolated pin. *Return type:* boolean For example, `get_property [get_lib_pins sc_cmoslp/AO22X/I1] is_isolated`
`is_isolation_cell_data`	Returns a value of `true` if the `isolation_cell_data_pin` attribute of a library signal pin is set to `true` indicating that the library cell signal pin is a data pin of an isolation library cell. *Return type:* boolean For example, `get_property [get_lib_pins sc_cmoslp/AO22X/I1] is_isolation_cell_data`
`is_isolation_cell_enable`	Returns a value of `true` if the `isolation_cell_enable_pin` attribute of a library signal pin is set to `true` indicating that the library cell signal pin is an enable or control pin of an isolation library cell. *Return type:* boolean For example, `get_property [get_lib_pins sc_cmoslp/AO22X/I1] is_isolation_cell_enable`
`is_level_shifter_data`	Returns a value of `true` if the `level_shifter_data_pin` attribute of a library signal pin is set to `true`, indicating that the library cell signal pin is a data pin of a library level-shifter cell. *Return type:* boolean For example, `get_property [get_lib_pins sc_cmoslp/AO22X/I1] is_level_shifter_data`
`is_level_shifter_enable`	Returns a value of `true` if the `level_shifter_enable_pin` attribute of a library signal pin is set to be `true` indicating that the library-cell signal pin is an enable pin of a library level shifter cell. *Return type:* boolean For example, `get_property [get_lib_pins sc_cmoslp/AO22X/I1] is_isolation_cell_enable`
`is_power_switch_enable`	Returns a value of `true` if the `switch_cell_type` attribute of the corresponding library cell is defined and the `switch_pin` attribute of the given library-cell signal pin is defined to be `true`. *Return type:* boolean For example, `get_property [get_lib_pins sc_cmoslp/AO22X/I1] is_power_switch_enable`

`is_retention_cell_enable`	Returns a value of `true` if the `retention_pin` attribute of a signal pin is defined. *Return type:* boolean For example, `get_property [get_lib_pins sc_cmoslp/AO22X/I1] is_retention_cell_enable`
`output_signal_level`	Returns the power rail name/signal that is defined for a library cell output and inout pins. *Return type:* string For example, `get_property [get_lib_pins sc_cmoslp/AO22X/I1] output_signal_level`
`output_signal_level_high`	Returns the upper/higher limit of partial voltage swing value that is defined for a library cell output and inout pins. *Return type:* float For example, `get_property [get_lib_pins sc_cmoslp/AO22X/I1] output_signal_level_high`
`output_signal_level_low`	Returns the lower limit of partial voltage swing value that is defined for a library cell output and inout pins. *Return type:* float For example, `get_property [get_lib_pins sc_cmoslp/AO22X/I1] output_signal_level_low`
`related_ground_pin`	Returns the related ground pins as a collection for the corresponding library cell pins. *Return type:* collection For example, `get_property [get_lib_pins sc_cmoslp/AO22X/I1] related_ground_pin`
`related_ground_pin_name`	Returns the related ground pin names for the corresponding library cell pins. *Return type:* string For example, `get_property [get_lib_pins sc_cmoslp/AO22X/I1] related_ground_pin_name`
`related_ground_pin_rail_voltage`	Returns the voltage value corresponding to the ground rail name defined for the `related_ground_pin` attribute of the library cell pin. This voltage value is the full rail-to-rail value that is associated with the relative ground rail name through the `voltage_map` attribute defined in the library. *Return type:* float For example, `get_property [get_lib_pins sc_cmoslp/AO22X/I1] related_ground_pin_rail_voltage`

`related_power_pin`	Returns the related power pins as a collection for the corresponding library cell pins. *Return type:* collection For example, `get_property [get_lib_pins sc_cmoslp/AO22X/I1] related_power_pin`
`related_power_pin_name`	Returns the related power pin names for the corresponding library cell pins. *Return type:* string For example, `get_property [get_lib_pins sc_cmoslp/AO22X/I1] related_power_pin_name`
`related_power_pin_rail_voltage`	Returns the voltage value corresponding to the power rail name defined for the `related_power_pin` attribute of a library cell pin. This voltage value is the full rail-to-rail value that is associated with the relative power rail name through the `voltage_map` attribute defined in the library. *Return type:* float For example, `get_property [get_lib_pins sc_cmoslp/AO22X/I1] related_power_pin_rail_voltage`

Library Pin Properties for Power/Ground Objects

`is_std_cell_main_rail`	Returns a value of true if the library power pg_pin is the main rail of the library cell. It is determined by the std_cell_main_rail library attribute of library pg_pin. *Return type*: boolean For example, `get_property [get_lib_pg_pins -of_objects [get_lib_cells sc_cmoslp/AO22X]] is_std_cell_main_rail`
`name`	Returns the name of the library pg pin as defined in the library and associated with the pg_pin library attribute. *Return type*: string For example, `get_property [get_lib_pg_pins -of_objects [get_lib_cells sc_cmoslp/AO22X]] name`
`object_type`	Returns lib_pg_pin as the object type for a library pg pin. *Return type*: string For example, `get_property [get_lib_pg_pins -of_objects [get_lib_cells sc_cmoslp/AO22X]] object_type`

`pg_type`	Returns the type of the library pg pin as associated with the pg_type attribute of the corresponding library pg pin.
	Return type: string
	
	For example,
	`get_property [get_lib_pg_pins -of_objects [get_lib_cells sc_cmoslp/AO22X]] pg_type`
	`get_property [get_lib_pg_pins -of_objects [get_lib_cells sc_cmoslp/AO22X] -filter " name =~ *VDD*" ] pg_type`

`rail_name`	Returns the name of the rail to which the pg-pin is connected and is associated with the voltage_name attribute of the library pg-pin. *Return type*: string For example, `get_property [get_lib_pg_pins -of_objects [get_lib_cells sc_cmoslp/AO22X]] rail_name`
`rail_voltage`	Returns the voltage from the voltage_map association of the rail to which the library pg-pin is connected. *Return type*: float For example, `get_property [get_lib_pg_pins -of_objects [get_lib_cells sc_cmoslp/AO22X]] rail_voltage`
`tied_to`	Returns the library pg pin which is connected to substrate-bias pg pin. *Return type*: string For example, `get_property [get_lib_pg_pins -of_objects [get_lib_cells sc_cmoslp/AO22X]] tied_to`

Library Timing Arc Properties

`from_lib_pin`	Returns a collection of the source library pins of the library timing arc. *Return type:* collection
`is_disabled`	Return a value of `true` if a `set_disable_timing` constraint exists for a an arc. *Return type:* boolean
`object_type`	Returns the object type `lib_timing_arc`. *Return type:* string
`sdf_cond`	Returns the `sdf_cond` expression specified for the timing library arc in the library. *Return type:* string
`sense`	Returns the `timing_sense` specified for the timing library arc in the library. *Return type:* string
`to_lib_pin`	Returns a collection of the sink library pins of the library timing arc. *Return type:* collection
`timing_type`	Returns the type specified for the timing library arc in the library. *Return type:* string
`when`	Returns the `when` condition specified for the library timing arc in the library. *Return type:* string

Net Properties

`attackers`	Reports the parasitic data on a net - capacitance, resistance, or cross-coupling. *Return type:* collection
`capacitance_max`	Returns the total capacitance of the net for maximum conditions. *Return type:* float
`coupling_capacitance_max`	Returns the maximum value of coupling capacitance of a net over all the views. *Return type:* float
`capacitance_min`	Returns the total capacitance of the net for minimum conditions. *Return type:* float

`coupling_capacitance_min`	Returns the minimum value of coupling capacitance of a net over all the views. *Return type:* float
`driver_pins`	Returns a collection of the driver pins of the net. *Return type:* collection
`early_fall_clk_net_derate_factor`	Returns the early fall clock net derating factor specified through `set_timing_derate` command. *Return type:* float
`early_fall_clk_net_delta_derate_factor`	Returns the early fall clock delta delay net derating factor specified through the `set_timing_derate` command. *Return type:* float
`early_fall_data_net_delta_derate_factor`	Returns the early fall data delta net derating factor specified through the `set_timing_derate` command. *Return type:* float
`early_fall_data_net_derate_factor`	Returns the early fall data net derating factor specified through `set_timing_derate` command. *Return type:* float
`early_rise_clk_net_delta_derate_factor`	Returns the early rise clock delta net derating factor specified through the `set_timing_derate` command. *Return type:* float
`early_rise_clk_net_derate_factor`	Returns the early rise clock net derating factor specified through the `set_timing_derate` command. *Return type:* float
`early_rise_data_net_delta_derate_factor`	Returns the early rise data delta net derating factor specified through the `set_timing_derate` command. *Return type:* float
`early_rise_data_net_derate_factor`	Returns the early rise data net derating factor specified through the `set_timing_derate` command. *Return type:* float
`has_detailed_parasitics`	Returns a value of `true` if the net, or one of its parts, has detailed parasitics associated with it from either RC extraction or SPEF annotation. *Return type:* boolean

`hierarchical_name`	Returns the complete hierarchical name of the net. *Return type:* string
`is_ideal`	Returns a value of true if `set_ideal_latency`, `set_ideal_transition`, or `set_ideal_network` constraints exist for the net. *Return type*: Boolean
`late_fall_clk_net_delta_derate_factor`	Returns the late fall clock delta net derating factor specified through the `set_timing_derate` command. *Return type:* float
`late_fall_clk_net_derate_factor`	Returns the late fall clock net derating factor specified through the `set_timing_derate` command. *Return type:* float
`late_fall_data_net_delta_derate_factor`	Returns the late fall data delta net derating factor specified through the `set_timing_derate` command. *Return type:* float
`late_fall_data_net_derate_factor`	Returns the late fall data net derating factor specified through the `set_timing_derate` command. *Return type:* float
`late_rise_clk_net_delta_derate_factor`	Returns the late rise clock delta net derating factor specified through the `set_timing_derate` command. *Return type:* float
`late_rise_clk_net_derate_factor`	Returns the late rise clock net derating factor specified through the `set_timing_derate` command. *Return type:* float
`late_rise_data_net_delta_derate_factor`	Returns the late rise data delta net derating factor specified through the `set_timing_derate` command. *Return type:* float
`late_rise_data_net_derate_factor`	Returns the late rise data net derating factor specified through the `set_timing_derate` command. *Return type:* float
`is_dont_touch`	Returns a value of `true` if a `set_dont_touch` constraint exists for the net. *Return type:* boolean
`load_pins`	Returns a collection of the load pins of the net. *Return type:* collection

`lumped_capacitance_max`	Returns the maximum value of capacitance of a net based on the input library and parasitic model. *Return type:* float
`lumped_capacitance_min`	Returns the minimum value of capacitance of a net based on the input library and parasitic model. *Return type:* float
`name`	Returns the name of the net. *Return type:* string
`num_driver_pins`	Returns the number of driver pins on a net. *Return type*: integer
`num_load_pins`	Returns the number of load pins on the net. *Return type*: collection
`object_type`	Returns the object type `net`. *Return type:* string
`pin_capacitance_max`	Returns the sum of all pin capacitances of the net for maximum conditions. *Return type:* float
`pin_capacitance_max_fall`	Returns the sum of all pin capacitances of the net for the maximum value of the rising capacitance range. *Return type:* float
`pin_capacitance_max_rise`	Returns the sum of all pin capacitances of the net for the maximum value of the falling capacitance range. *Return type:* float
`pin_capacitance_min`	Returns the sum of all pin capacitances of the net for minimum conditions. *Return type:* float
	Note: For all the `pin_capacitance_*` properties (described above), the pin capacitance matches with the capacitance displayed under the `pin_load` column of the `report_timing` output.
`resistance_max`	Returns the maximum value of resistance of a net. *Return type*: float
`resistance_min`	Returns the minimum value of resistance of a net. *Return type*: float

`total_lumped_capacitance_max_fall`	Returns the maximum value of falling capacitance of a net based on the input library and parasitic model. *Return type:* float
`total_lumped_capacitance_max_rise`	Returns the maximum value of rising capacitance of a net based on the input library and parasitic model. *Return type:* float
`total_lumped_capacitance_min_fall`	Returns the minimum value of falling capacitance of a net based on the input library and parasitic model. *Return type:* float
`total_lumped_capacitance_min_rise`	Returns the minimum value of rising capacitance of a net based on the input library and parasitic model. *Return type:* float
`wire_capacitance_max`	Returns the wire capacitance of the net for maximum conditions. *Return type:* float
`wire_capacitance_min`	Returns the wire capacitance of the net for minimum conditions. *Return type:* float

Net Properties for Power/Ground Objects

`hierarchical_name`	Returns the name of the PG net connected to the corresponding PG pin.
	Return type: string
	
	For example,
	`get_property [get_pg_nets -of_objects [get_cells iA/out_reg] ] hierarchical_name`
	`get_property [get_pg_nets -of_objects [get_cells iA/out_reg] -filter "is_power == true" ] hierarchical_name`

`is_ground`	Returns a value of `true` if the PG net is a ground net.
	Return type: boolean
	
	For example,
	`get_property [get_pg_nets -of_objects [get_cells iA/out_reg] ] is_ground`
	`get_property [get_pg_nets -of_objects [get_pg_pins -of_objects [get_cells iA/out_reg] ]] is_ground`

`is_power`	Returns a value of `true` if the PG net is a power net.
	Return type: boolean
	
	For example,
	`get_property [get_pg_nets -of_objects [get_cells iA/out_reg] ] is_power`
	`get_property [get_pg_nets -of_objects [get_cells iA/out_reg] -filter "name =~ *VDD*" ] is_power`

`voltage_max`	Returns the maximum voltage of a given PG net. This could be voltage associated with a driver term of a net, power intent or CPF, `related_pg_pins` attribute in Liberty, or the operating voltage of a power domain or system default voltage. *Return type*: float For example, `get_property [get_pg_nets -of_objects [get_cells iA/out_reg] ] voltage_max`
`voltage_min`	Returns the minimum voltage of a given PG net. This could be voltage associated with a driver term of a net, power intent or CPF, `related_pg_pins` attribute in Liberty or the operating voltage of a power domain or system default voltage. *Return type*: float For example, `get_property [get_pg_nets -of_objects [get_cells iA/out_reg] ] voltage_min`
`object_type`	Returns `pg_net` as the object type for a library PG pin. *Return type*: string For example, `get_property [get_pg_nets -of_objects [get_cells iA/out_reg]] object_type`

`parent_pg_net`	Returns a collection of a parent power net that is the net powering the power switch that drives the PG net. In case of designs that do not have any power switches powering the instances, this property will always return the net directly powering those instances. It therefore be noted that the usage of this property is more relevant for designs in which the leaf cell instances are powered through power switch cells.
	Return type: Collection
	
	For example,
	`get_property [get_pg_nets mmu/U1/vdd] parent_pg_net`
	`get_property [get_pg_nets -of_objects [get_cells iA/pwr_swtch] ] parent_pg_net`

`top_pg_net`	Returns a collection of the top most un-switched power net of the current PG net. Even in the case of a cascading of power switches present in a design, the software traces through all power switches back to the un-switched top-level primary power net. The top-most un-switched power net will be returned - switched output power net will not be returned.
	Return type: Collection
	
	For example,
	`get_property [get_pg_nets -of_objects [get_cells iA/pwr_swtch] ] top_pg_net`
	`get_property [get_pg_nets -of_objects [get_pg_pins -of_objects [get_cells iA/pwr_swtch] ]] top_pg_net`

Path Group Properties

`name`	Returns the name of the path group. *Return type:* string
`hierarchical_name`	Returns the hierarchical name of the path group. *Return type:* string
`object_type`	Returns the object type `path_group`. *Return type:* string

Pin Properties

`actual_latency_early_fall_max`	When specified with the `set_clock_latency -clock` parameter, returns the worst actual early fall insertion delay in setup mode for the specified clock at the given pin. The latency reported is ideal or propagated, depending on the clock phase. When specified without the `-clock` parameter, returns the worst actual early fall insertion delay on the pin. If multiple clocks reach the pin, the reported latency is the worst among all the clocks. When specified with the `-view` parameter, returns the worst latency for the given view. *Return type:* float
`actual_latency_early_fall_min`	When specified with the `set_clock_latency -clock` parameter, returns the worst actual early fall insertion delay in hold mode for the specified clock at the given pin. The latency reported is ideal or propagated, depending on the clock phase. When specified without the `-clock` parameter, returns the worst actual early fall insertion delay on the pin. If multiple clocks reach the pin, the reported latency is the worst among all the clocks. When specified with the `-view` parameter, returns the worst latency for the given view. *Return type:* float

`actual_latency_early_rise_max`	When specified with the `set_clock_latency -clock` parameter, returns the worst actual early rise source insertion delay in setup mode for the specified clock at the given pin. The latency reported is ideal or propagated, depending on the clock phase. When specified without the `-clock` parameter, returns the worst actual early rise source insertion delay on the pin. If multiple clocks reach the pin, the reported latency is the worst among all the clocks. When specified with the `-view` parameter, returns the worst latency for the given view. *Return type:* float
`actual_latency_early_rise_min`	When specified with the `set_clock_latency -clock` parameter, returns the worst actual early rise source insertion delay in hold mode for the specified clock at the given pin. The latency reported is ideal or propagated, depending on the clock phase. When specified without the `-clock` parameter, returns the worst actual early rise source insertion delay on the pin. If multiple clocks reach the pin, the reported latency is the worst among all the clocks. When specified with the `-view` parameter, returns the worst latency for the given view. *Return type:* float
`actual_latency_late_fall_max`	When specified with the `set_clock_latency -clock` parameter, returns the worst actual late fall source insertion delay in setup mode for the specified clock at the given pin. The latency reported is ideal or propagated, depending on the clock phase. When specified without the `-clock` parameter, returns the worst actual late fall source insertion delay on the pin. If multiple clocks reach the pin, the reported latency is the worst among all the clocks. When specified with the `-view` parameter, returns the worst latency for the given view. *Return type:* float
`actual_latency_late_fall_min`	When specified with the `set_clock_latency -clock` parameter, returns the worst actual late fall source insertion delay in hold mode for the specified clock at the given pin. The latency reported is ideal or propagated, depending on the clock phase. When specified without the `-clock` parameter, returns the worst actual late fall source insertion delay on the pin. If multiple clocks reach the pin, the reported latency is the worst among all the clocks. When specified with the `-view` parameter, returns the worst latency for the given view. *Return type:* float

`actual_latency_late_rise_max`	When specified with the `set_clock_latency -clock` parameter, returns the worst actual late rise source insertion delay in setup mode for the specified clock at the given pin. The latency reported is ideal or propagated, depending on the clock phase. When specified without the `-clock` parameter, returns the worst actual late rise source insertion delay on the pin. If multiple clocks reach the pin, the reported latency is the worst among all the clocks. When specified with the `-view` parameter, returns the worst latency for the given view. *Return type:* float
`actual_latency_late_rise_min`	When specified with the `set_clock_latency -clock` parameter, returns the worst actual late rise source insertion delay in hold mode for the specified clock at the given pin. The latency reported is ideal or propagated, depending on the clock phase. When specified without the `-clock` parameter, returns the worst actual late rise source insertion delay on the pin. If multiple clocks reach the pin, the reported latency is the worst among all the clocks. When specified with the `-view` parameter, returns the worst latency for the given view. *Return type:* float
`arrival_clocks`	Returns the data phases, clock source data phases, and latency phases. *Return type:* collection The following attributes support these phases: • `timing_property_arrival_clocks_consider_clock_source_data_phase`: Considers the clock source data phases. • `timing_property_arrival_clocks_consider_data_phase`: Considers the pure data phase. • `timing_property_arrival_clocks_consider_latency_phase`: Considers the latency phase. By default, these attributes are set to `true`.
`arrival_max_fall`	Returns the arrival time for the falling transition at the max corner. *Return type:* float
`arrival_max_rise`	Returns the arrival time for the rising transition at the max corner. *Return type:* float

`arrival_mean_max_fall`	Returns the mean component of the arrival time for the falling transition at the max corner. *Return type:* float
`arrival_mean_max_rise`	Returns the mean component of the arrival time for the rising transition at the max corner. *Return type:* float
`arrival_mean_min_fall`	Returns the mean component of the arrival time for the falling transition at the min corner. *Return type:* float
`arrival_mean_min_rise`	Returns the mean component of the arrival time for the rising transition at the min corner. *Return type:* float
`arrival_min_fall`	Returns the arrival time for the falling transition at the min corner. *Return type:* float
`arrival_min_rise`	Returns the arrival time for the rising transition at the min corner. *Return type:* float
`arrival_sigma_max_fall`	Returns the sigma component of the arrival time for the falling transition at the max corner. *Return type:* float
`arrival_sigma_max_rise`	Returns the mean component of the arrival time for the rising transition at the max corner. *Return type:* float
`arrival_sigma_min_fall`	Returns the sigma component of the arrival time for the falling transition at the min corner. *Return type:* float
`arrival_sigma_min_rise`	Returns the sigma component of the arrival time for the falling transition at the min corner. *Return type:* float
`arrival_window`	Returns the arrival times for the earliest and latest rising and falling transitions in timing window format. *Return type:* string

`attacker_fall_slew`	Returns the minimum falling slew of an attacker driver pin. For example: `get_property [get_pins inst1/y] attacker_fall_slew` *Return type:* float
`attacker_rise_slew`	Returns the minimum rising slew of an attacker driver pin. For example: `get_property [get_pins inst1/y] attacker_rise_slew` *Return type:* float
`capacitance_max_fall`	Returns the maximum value of the falling capacitance range of the corresponding library pin. *Return type:* float
`capacitance_max_rise`	Returns the maximum value of the rising capacitance range of the corresponding library pin. *Return type:* float
`capacitance_min_fall`	Returns the minimum value of the falling capacitance range of the corresponding library pin. *Return type:* float
`capacitance_min_rise`	Returns the minimum value of the rising capacitance range of the corresponding library pin. **Note**: For the `capacitance_max_*` and `capacitance_min_*` properties (described above), the capacitance value does not match with the `pin_load` (`report_timing -format`) in the timing report because the pin load is a sum of all pin capacitances of a net. *Return type:* float
`clock_sources`	Returns a collection of the source pins of all the clocks arriving at the pin. *Return type:* collection
`clocks`	Returns a collection of all of the clocks arriving at the pin. *Return type:* collection
`constant_value`	Returns a constant value of `0` or `1`. *Return type:* integer
`delay_max_fall`	Returns the maximum falling delay - specified using the `set_max_delay -fall` command. *Return type:* float

`delay_max_rise`	Returns the maximum rising delay - specified using the `set_max_delay -rise` command. *Return type:* float
`delay_min_fall`	Returns the minimum falling delay - specified using the `set_min_delay -fall` command. *Return type:* float
`delay_min_rise`	Returns the minimum rising delay - specified using the `set_min_delay -rise` command. *Return type:* float
`direction`	Returns the direction of the pin: `in`, `out`, or `inout`. *Return type:* string
`fanin`	Returns the number of sources connected with net of the specified pin. *Return type:* integer
`fanout`	Returns the number of sinks connected with the net of the specified pin. *Return type:* integer
`fanout_load`	Returns the fanout load for the pin. *Return type:* float
`hierarchical_name`	Returns the complete hierarchical name of the pin. *Return type:* string
`hold_uncertainty`	Returns the most conservative uncertainty of all possible uncertainty assertions associated with the pin. *Return type:* float

`holding_resistance_high`	Returns the highest holding resistance of a pin.
	For example,
	`foreach pin $pinList {`
	`set vicArray [get_property [get_pins $pin] noise_victim]`
	`foreach_in_collection vic $vicArray {`
	`get_property $vic holding_resistance_high`
	`}`
	Return type: float

`holding_resistance_low`	Returns the lowest holding resistance of a pin. *Return type:* float

`interp_info_max`	Returns information for late analysis. When specified, the report output will show the data for interpolation, extrapolation, and/or matched information for late analysis, along with the library details. *Return type:* string
`interp_info_min`	Returns information for early analysis. When specified, the report output will show the data for interpolation, extrapolation, and/or matched information for early analysis, along with the library details. *Return type:* string
`is_async`	Returns a value of `true` if the pin is an asynchronous clear or preset pin. *Return type:* boolean
`is_clear`	Returns a value of `true` if the pin is an asynchronous clear pin. *Return type:* boolean
`is_clock`	Returns a value of `true` if the pin has the Liberty pin attribute: `clock`. *Return type:* boolean
`is_clock_gating`	Returns a value of `true` if the pin is defined as a pin of a clock gating cell. *Return type:* boolean
`is_clock_gating_clock`	Returns a value of `true` if the pin is defined as a clock pin of a clock gating cell. *Return type:* boolean
`is_clock_gating_enable`	Returns a value of `true` if the pin corresponds to the enable pin of a clock gating cell. *Return type:* boolean
`is_clock_used_as_clock`	Returns a value of `true` if the pin lies in the clock network and at least one of the clocks arriving at the pin is used as a clock in the downstream network of the pin. When the `-clock` parameter is specified, this property returns `true` if the above condition is satisfied for the specified clock. *Return type*: boolean

`is_clock_used_as_data`	Returns a value of `true` if the pin lies in the clock source path and at least one of the clocks arriving on the pin is used as data in the downstream network of the pin. When the `-clock` parameter is specified, this property returns `true` if the above condition is satisfied for the specified clock. *Return type*: boolean
`is_data`	Returns a value of `true` if the pin is a data pin of a flip-flop (that is, is not a clock pin). *Return type:* boolean
`is_disable_timing`	Returns a value of `true` if the pin's timing has been disabled. Return type: boolean **Note**: This property checks if the associated arc for the pin is active for propagation and analysis.
`is_fall_edge_triggered_clock`	Returns a value of `true` if the pin is a clock pin of a flop, and is triggered by the falling edge of a clock. *Return type:* boolean
`is_fall_edge_triggered_data`	Returns a value of `true` if the pin corresponds to the data pin of a fall edge triggered device. *Return value:* boolean
`is_hierarchical`	Returns a value of `true` if the pin is a pin of a hierarchical module. *Return type:* boolean
`is_isolated`	Returns a value of `true` when a pin is connected to an isolated instance. The `is_isolation` attribute indicates that the isolation cell is inside a module. So if the module is used as macro or black-box, the software will interpret the pin to be already isolated and no additional isolation will be required. *Return type:* boolean
`is_multiple_clock_fanin_point`	Returns a value of `true` if multiple clock phases converge at the pin. *Return type:* boolean
`is_negative_level_sensitive_clock`	Returns a value of `true` if the library pin is an enable pin of an active low level-sensitive device. *Return type:* boolean
`is_negative_level_sensitive_data`	Returns a value of `true` if the pin is a data pin of an active low level-sensitive device. *Return type:* boolean

`is_pin`	Returns a value of `true`. The `is_pin` and `is_port` queries are provided for both pin and port objects to provide a quick way of determining the object type of an identifier. `is_port` always returns `false` for pins. `is_pins` always returns `false` for ports. *Return type:* boolean
`is_port`	Returns a value of `false`. The `is_pin` and `is_port` queries are provided for both pin and port objects to provide a quick way of determining the object type of an identifier. `is_port` always returns `false` for pins. `is_pins` always returns `false` for ports. *Return type:* boolean
`is_positive_level_sensitive_clock`	Returns a value of `true` if the library pin is an enable pin of an active high level-sensitive device. *Return type:* boolean
`is_positive_level_sensitive_data`	Returns a value of `true` if the pin is a data pin of an active high level-sensitive device. *Return type:* boolean
`is_preset`	Returns a value of `true` if the pin is a preset pin. *Return type:* boolean
`is_propagated_clock`	Returns a value of `true` if there is an explicit `set_propagated_clock` assertion at the pin. *Return type:* boolean
`is_rise_edge_triggered_clock`	Returns a value of `true` if the pin is a clock pin of a flop, and is triggered by the rising edge of a clock. *Return type:* boolean
`is_rise_edge_triggered_data`	Returns a value of `true` if the pin corresponds to the data pin of a rise edge-triggered device. *Return type:* boolean
`is_tristate`	Returns a value of `true` if the pin has the `three_state` attribute in the Liberty timing library. *Return type:* boolean

`is_tristate_enable`	Returns a value of `true` if the pin is the source pin of timing arcs with either the `three_state_enable` or `three_state_disable` attribute in the Liberty timing library. *Return type:* boolean
`is_tristate_output`	Returns a value of `true` if the pin corresponds to a three-state output pin. *Return type:* boolean
`latency_fall_max`	Returns the maximum fall insertion delay specified by an explicit `set_clock_latency` at the pin. *Return type:* float
`latency_fall_min`	Returns the minimum fall insertion delay specified by an explicit `set_clock_latency` at the pin. *Return type:* float
`latency_rise_max`	Returns the maximum rise insertion delay specified by an explicit `set_clock_latency` at the pin. *Return type:* float
`latency_rise_min`	Returns the minimum rise insertion delay specified by an explicit `set_clock_latency` at the pin. *Return type:* float
`max_capacitance`	Returns the maximum capacitance limit for the pin. *Return type:* float
`max_fanout`	Returns the maximum fanout load that the pin can drive. This value is set using the `set_max_fanout` command or the `default_max_fanout` library attribute. *Return type:* float
`max_transition`	Returns the maximum transition time limit specified for the pin. *Return type:* float
`min_fanout`	Returns the minimum fanout design rule limit of the corresponding library pin. *Return type:* float
`net_name`	Returns the name of the net connected to the specified pin. *Return type:* string
`noise_victim`	Returns a collection of `si_victim` objects that can be used to query glitch attributes. *Return type:* collection

`object_type`	Returns the object type `pin`. *Return type:* string
`power_rail_voltage_bidir_input_max`	Reports power rail voltage for input part of bidi pins/ports for maximum operating condition. *Return type:* float
`power_rail_voltage_bidir_input_min`	Reports power rail voltage for input part of bidi pins/ports for minimum operating condition. *Return type:* float
`power_rail_voltage_max`	Reports power rail voltage for maximum operating condition for bidirectional, input and output pins. For bidirectional pins, the power rail voltage for output signal is reported. *Return type:* float
`power_rail_voltage_min`	Reports power rail voltage for minimum operating condition for bidirectional, input and output pins. For bidirectional pins, the power rail voltage for output signal is reported. *Return type:* float
`receiver_ccc_type`	Returns the type of receiver cell model. The return values can be udn, single_ccc, dual_ccc, multi_ccc, or black_box. This attribute applies to receiver pins only. For example, `get_property [get_pins inst1/a] receiver_ccc_type` *Return type:* string
`ref_lib_pin_name`	Returns the name of the library pin associated with the instance pin. *Return type:* string
`setup_uncertainty`	Returns the most conservative uncertainty of all possible uncertainty assertions associated with the pin. *Return type:* float
`si_victim`	Returns a collection of the `si_victim` objects that can be used to query graph-based SI delay attributes. *Return type:* collection
`slack_max_edge`	Returns the edge (rise or fall) of the worst slack-causing path at the specified pin in late mode. *Return type:* string
`slack_max_fall`	Returns the slack time for the falling transition at the max corner. *Return type:* float

`slack_max_rise`	Returns the slack time for the rising transition at the max corner. *Return type:* float
`slack_mean_max_fall`	Returns the mean component of the slack time for the falling transition at the max corner. *Return type:* float
`slack_mean_max_rise`	Returns the mean component of the slack time for the rising transition at the max corner. *Return type:* float
`slack_mean_min_fall`	Returns the mean component of the slack time for the falling transition at the min corner. *Return type:* float
`slack_mean_min_rise`	Returns the mean component of the slack time for the rising transition at the min corner. Return type: float
`slack_min_edge`	Returns the edge (rise or fall) of the worst slack-causing path at the specified pin in early mode. *Return type:* string
`slack_min_fall`	Returns the slack time for the falling transition at the min corner. *Return type:* float
`slack_min_rise`	Returns the slack time for the rising transition at the min corner. *Return type:* float
`slack_sigma_max_fall`	Returns the sigma component of the slack time for the falling transition at the max corner. *Return type:* float
`slack_sigma_max_rise`	Returns the sigma component of the slack time for the rising transition at the max corner. *Return type:* float
`slack_sigma_min_fall`	Returns the sigma component of the slack time for the falling transition at the min corner. *Return type:* float
`slack_sigma_min_rise`	Returns the sigma component of the slack time for the rising transition at the min corner. *Return type:* float

`slew_max_fall`	Returns the slew time for the falling transition at the max corner. *Return type:* float
`slew_max_rise`	Returns the slew time for the rising transition at the max corner. *Return type:* float
`slew_mean_max_fall`	Returns the mean component of the slew time for the falling transition at the max corner. *Return type:* float
`slew_mean_max_rise`	Returns the mean component of the slew time for the rising transition at the max corner. *Return type:* float
`slew_mean_min_fall`	Returns the mean component of the slew time for the falling transition at the min corner. *Return type:* float
`slew_mean_min_rise`	Returns the mean component of the slew time for the rising transition at the min corner. *Return type:* float
`slew_min_fall`	Returns the slew time for the falling transition at the min corner. *Return type:* float
`slew_min_rise`	Returns the slew time for the rising transition at the min corner. *Return type:* float
`slew_sigma_max_fall`	Returns the sigma component of the slew time for the falling transition at the max corner. *Return type:* float
`slew_sigma_max_rise`	Returns the sigma component of the slew time for the rising transition at the max corner. *Return type:* float
`slew_sigma_min_fall`	Returns the sigma component of the slew time for the falling transition at the min corner. *Return type:* float
`slew_sigma_min_rise`	Returns the sigma component of the slew time for the falling transition at the min corner. *Return type:* float

`source_latency_early_fall_max`	Returns the maximum early fall source insertion delay specified by an explicit `set_clock_latency` at the pin. *Return type:* float
`source_latency_early_fall_min`	Returns the minimum early fall source insertion delay specified by an explicit `set_clock_latency` at the pin. *Return type:* float
`source_latency_early_rise_max`	Returns the maximum early rise source insertion delay specified by an explicit `set_clock_latency` at the pin. *Return type:* float
`source_latency_early_rise_min`	Returns the minimum early rise source insertion delay specified by an explicit `set_clock_latency` at the pin. *Return type:* float
`source_latency_late_fall_max`	Returns the maximum late fall source insertion delay specified by an explicit `set_clock_latency` at the pin. *Return type:* float
`source_latency_late_fall_min`	Returns the minimum late fall source insertion delay specified by an explicit `set_clock_latency` at the pin. *Return type:* float
`source_latency_late_rise_max`	Returns the maximum late rise source insertion delay specified by an explicit `set_clock_latency` at the pin. *Return type:* float
`source_latency_late_rise_min`	Returns the minimum late rise source insertion delay specified by an explicit `set_clock_latency` at the pin. *Return type:* float
`stolen_slack`	Returns the slack stolen at the specified pin. This property can be queried only for pins in late mode. **Note:** The specified pin must be the endpoint of a transparent latch flush arc. *Return type:* float
`user_constant_value`	Returns constant values from netlist or constraints. *Return type*: integer
`x_coordinate`	Returns the x coordinates of the placed pin. *Return type:* float

`y_coordinate`	Returns the y coordinates of the placed pin. *Return type:* float

Pin Properties for Power/Ground Objects	
`input_signal_level`	Returns the rail name of the `related_power_pin` of an input signal pin. *Return type:* string For example, `get_property [get_pins mmu/U1/I] input_signal_level`
`input_signal_level_voltage`	Returns the rail voltage of a rail of the `related_power_pin` attribute of the input signal pin. *Return type:* float For example, `get_property [get_pins mmu/U1/I] input_signal_level_voltage`
`is_always_on`	Returns a value of `true` if an instance pin is an always-on pin. *Return type:* boolean For example, `get_property [get_pins mmu/U1/AON] is_always_on`
`is_isolation_cell_data`	Returns a value of `true` if the instance signal pin is the data pin of an isolation cell instance. *Return type:* boolean For example, `get_property [get_pins mmu/U2/ISO] is_isolation_cell_data`
`is_isolation_cell_enable`	Returns a value of `true` if the instance signal pin is the enable pin of an isolation cell instance. *Return type:* boolean For example, `get_property [get_pins mmu/U2/ISO_EN] is_isolation_cell_enable`
`is_level_shifter_data`	Returns a value of `true` if the instance signal pin is the data pin of a level-shifter cell instance. *Return type:* boolean For example, `get_property [get_pins mmu/LVL/IN] is_level_shifter_data`

`is_level_shifter_enable`	Returns a value of `true` if the instance signal pin is the enable pin of a level-shifter cell instance. *Return type:* boolean For example, `get_property [get_pins mmu/LVL/EN] is_level_shifter_enable`
`output_signal_level`	Returns the rail name of the `related_power_pin` attribute of the output/inout signal pin. *Return type:* string For example, `get_property [get_pins mmu/U1/Y] output_signal_level`
`related_ground_pin`	Returns the related ground pins as a collection for the corresponding instance pins. *Return type:* collection For example, `get_property [get_pins mmu/U1/I] related_ground_pin`
`related_power_pin`	Returns the related power pins as a collection for the corresponding instance pins. *Return type:* collection For example, `get_property [get_pins mmu/U1/I] related_power_pin`

Pin Properties for Power/Ground Objects

`hierarchical_name`	Returns the complete hierarchical name of the PG pin.
	Return type: string
	
	For example,
	`get_property [get_pg_pins -of_objects [get_cells iA/out_reg] ] hierarchical_name`
	`get_property [get_pg_pins -of_objects [get_cells iA/out_reg] -filter "pg_type == primary_power" ] hierarchical_name`

`voltage_max`	Returns the maximum voltage of a given PG pin. This could be the voltage associated with a pin, power intent or CPF, `related_pg_pins` attribute in Liberty, or the operating voltage of a power domain or system default voltage. *Return type:* float For example, `get_property [get_pg_pins -of_objects [get_cells iA/out_reg]] voltage_max`

`voltage_min`	Returns the minimum voltage of a given PG pin. This voltage may be associated with a pin, power intent or CPF, `related_pg_pins` attribute in Liberty, or the operating voltage of a power domain or system default voltage. *Return type*: float For example, `get_property [get_pg_pins -of_objects [get_cells iA/out_reg]] voltage_min`
`net`	Returns the immediate PG net(s) connected to the corresponding PG pin objects, as a collection. *Return type*: Collection For example, `get_property [get_pg_pins -of_objects [get_cells iA/out_reg]] net`

`pg_type`	Returns the type of PG pin.
	Return type: string
	
	For example,
	`get_property [get_pg_pins -of_objects [get_cells iA/out_reg]] pg_type`
	`get_property [get_pg_pins -of_objects [get_cells iA/out_reg] -filter "hierarchical_name =~ *VDD*" ] pg_type`

`ref_lib_pin_name`	Returns the name of the library PG pin associated with an instance PG pin. *Return type*: string For example, `get_property [get_pg_pins -of_objects [get_cells iA/out_reg]] ref_lib_pin_name`

Port Properties

`arrival_max_fall`	Returns the worst arrival on the port for a falling signal. *Return type:* float
`arrival_max_rise`	Returns the worst arrival on the port for a rising signal. *Return type:* float
`arrival_min_fall`	Returns the best arrival on the port for a falling signal. *Return type:* float
`arrival_min_rise`	Returns the best arrival on the port for a rising signal. *Return type:* float
`arrival_window`	Returns the arrival times for the earliest and latest rising and falling transitions in timing window format. *Return type:* string

`clock_sources`	Returns a collection of the source pins of all the clocks arriving at the port. *Return type:* collection
`clocks`	Returns a collection of all of the clocks arriving at the port. *Return type:* collection
`constant_value`	Returns a constant value of `0` or `1`. *Return type:* integer
`capacitance_max`	Returns the maximum pin capacitance for the port. *Return type:* float
`capacitance_min`	Returns the minimum pin capacitance for the port. *Return type:* float
`delay_max_fall`	Returns the maximum falling delay value for the port. *Return type:* float
`delay_max_rise`	Returns the maximum rising delay value for the port. *Return type:* float
`delay_min_fall`	Returns the minimum falling delay value for the port. *Return type:* float
`delay_min_rise`	Returns the minimum rising delay value for the port. *Return type:* float
`direction`	Returns the direction of the port: `in`, `out`, or `inout`. *Return type:* string
`drive_resistance_fall_max`	Returns the falling linear drive resistance at the port for the max condition. *Return type:* float
`drive_resistance_fall_min`	Returns the falling linear drive resistance at the port for the min condition. *Return type:* float
`drive_resistance_rise_max`	Returns the rising linear drive resistance at the port for the max condition. *Return type:* float
`drive_resistance_rise_min`	Returns the rising linear drive resistance at the port for the min condition. *Return type:* float
`driving_cell_fall_max`	Returns the name of the library cell used to compute max falling delays and transitions at the port. *Return type:* string

`driving_cell_fall_min`	Returns the name of the library cell used to compute min falling delays and transitions at the port. *Return type:* string
`driving_cell_from_pin_fall_max`	Returns the input pin of the driving cell used in the max fall delay calculation at the port. *Return type:* string
`driving_cell_from_pin_fall_min`	Returns the input pin of the driving cell used in the min fall delay calculation at the port. *Return type:* string
`driving_cell_from_pin_rise_max`	Returns the input pin of the driving cell used in max rise delay calculation at the port. *Return type:* string
`driving_cell_from_pin_rise_min`	Returns the input pin of the driving cell used in min rise delay calculation at the port. *Return type:* string
`driving_cell_library_fall_max`	Returns the name of the library used to compute max falling delays and transitions at the port. *Return type:* string
`driving_cell_library_fall_min`	Returns the name of the library used to compute min falling delays and transitions at the port. *Return type:* string
`driving_cell_library_rise_max`	Returns the name of the library used to compute max rising delays and transitions at the port. *Return type:* string
`driving_cell_library_rise_min`	Returns the name of the library used to compute min rising delays and transitions at the port. *Return type:* string
`driving_cell_max_fall_itrans_fall`	Returns the falling input transition of the `from_pin` of the driving cell, which is used for the max falling delay calculation at the port. *Return type:* float
`driving_cell_max_fall_itrans_rise`	Returns the falling input transition of the `from_pin` of the driving cell, which is used for the max rising delay calculation at the port. *Return type:* float

`driving_cell_max_rise_itrans_fall`	Returns the rising input transition of the `from_pin` of the driving cell, which is used for the max falling delay calculation at the port. *Return type:* float
`driving_cell_max_rise_itrans_rise`	Returns the rising input transition of the `from_pin` of the driving cell, which is used for the max rising delay calculation at the port. *Return type:* float
`driving_cell_min_fall_itrans_fall`	Returns the falling input transition of the `from_pin` of the driving cell, which is used for the min falling delay calculation at the port. *Return type:* float
`driving_cell_min_fall_itrans_rise`	Returns the falling input transition of the `from_pin` of the driving cell, which is used for the min rising delay calculation at the port. *Return type:* float
`driving_cell_min_rise_itrans_fall`	Returns the rising input transition of the `from_pin` of the driving cell, which is used for the min falling delay calculation at the port. *Return type:* float
`driving_cell_min_rise_itrans_rise`	Returns the rising input transition of the `from_pin` of the driving cell, which is used for the min rising delay calculation at the port. *Return type:* float
`driving_cell_pin_fall_max`	Returns the name of the output pin used for max fall delay calculation at the port. *Return type:* string
`driving_cell_pin_fall_min`	Returns the name of the output pin used for min fall delay calculation at the port. *Return type:* string
`driving_cell_pin_rise_max`	Returns the name of the output pin used for max rise delay calculation at the port. *Return type:* string
`driving_cell_pin_rise_min`	Returns the name of the output pin used for min rise delay calculation at the port. *Return type:* string
`driving_cell_rise_max`	Returns the name of the library cell used to compute max rising delays and transitions at the port. *Return type:* string

`driving_cell_rise_min`	Returns the name of the library cell used to compute min rising delays and transitions at the port. *Return type:* string
`fanin`	Returns the number of sources connected with net of the specified port. *Return type:* integer
`fanout`	Returns the number of sinks connected with the net of the specified port. *Return type:* integer
`fanout_load`	Returns the fanout load for ports. (For input ports, the value is the sum of all fanout loads of all connected pins. For output ports, the value is set using `set_fanout_load`.) *Return type:* float
`hierarchical_name`	Returns the complete hierarchical name of the port. *Return type:* string
`hold_uncertainty`	Returns the most conservative uncertainty of all possible uncertainty assertions associated with the port. *Return type:* float
`input_slew_max_fall`	Returns the max falling slew for the input or inout ports. (This value is set with the `set_input_transition` command.) *Return type:* float
`input_slew_max_rise`	Returns the max rising slew for the input or inout ports. (This value is set with the `set_input_transition` command.) *Return type:* float
`input_slew_min_fall`	Returns the min falling slew for the input or inout ports. (This value is set with the `set_input_transition` command.) *Return type:* float
`input_slew_min_rise`	Returns the min rising slew for the input or inout ports. (This value is set with the `set_input_transition` command.) *Return type:* float
`is_disable_timing`	Returns a value of `true` if the timing of the port has been disabled using the `set_disable_timing` command. *Return type:* boolean
`is_hierarchical`	Returns a value of `true` if the port is a port of a hierarchical module. *Return type:* boolean

`is_pin`	Returns a value of `false`. The `is_pin` and `is_port` queries are provided for both pin and port objects to provide a quick way of determining the object type of an identifier. `is_port` always returns `false` for pins. `is_pins` always returns `false` for ports. *Return type:* boolean
`is_port`	Returns a value of `true`. The `is_pin` and `is_port` queries are provided for both pin and port objects to provide a quick way of determining the object type of an identifier. `is_port` always returns `false` for pins. `is_pins` always returns `false` for ports. *Return type:* boolean
`is_propagated_clock`	Returns a value of `true` if there is an explicit `set_propagated_clock` assertion at the port. *Return type:* boolean
`latency_fall_max`	Returns the maximum fall insertion delay specified by an explicit `set_clock_latency` at the port. *Return type:* float
`latency_fall_min`	Returns the minimum fall insertion delay specified by an explicit `set_clock_latency` at the port. *Return type:* float
`latency_rise_max`	Returns the maximum rise insertion delay specified by an explicit `set_clock_latency` at the port. *Return type:* float
`latency_rise_min`	Returns the minimum rise insertion delay specified by an explicit `set_clock_latency` at the port. *Return type:* float
`max_fanout`	Returns the maximum fanout load that the port can drive. (This value is set using `set_max_fanout`, or the `default_max_fanout` library attribute.) *Return type:* float
`max_transition`	Returns the maximum transition time limit specified for the port. *Return type:* float
`net_name`	Returns the name of the net connected to the specified port. *Return type:* string

`noise_victim`	Returns a collection of `si_victim` objects that can be used to query glitch attributes. *Return type:* collection
`object_type`	Returns the object type `port`. *Return type:* string
`setup_uncertainty`	Returns the most conservative uncertainty of all possible uncertainty assertions associated with the port. *Return type:* float
`si_victim`	Returns a collection of the `si_victim` objects that can be used to query graph-based SI delay attributes. *Return type:* collection
`slack_max_edge`	Returns the edge (rise or fall) of the worst slack-causing path at the specified port in late mode. *Return type:* string
`slack_max_fall`	Returns the worst setup slack for a falling signal at the port endpoint. *Return type:* float
`slack_max_rise`	Returns the worst setup slack for a rising signal at the port endpoint. *Return type:* float
`slack_min_edge`	Returns the edge (rise or fall) of the worst slack-causing path at the specified port in early mode. *Return type:* string
`slack_min_fall`	Returns the worst hold slack for a falling signal at the port endpoint. *Return type:* float
`slack_min_rise`	Returns the worst hold slack for a rising signal at the port endpoint. *Return type:* float
`slew_max_fall`	Returns the maximum falling slew from all of the incoming signals on the port. *Return type:* float
`slew_max_rise`	Returns the maximum rising slew from all of the incoming signals on the port. *Return type:* float
`slew_min_fall`	Returns the minimum falling slew from all of the incoming signals on the port. *Return type:* float

`slew_min_rise`	Returns the minimum rising slew from all of the incoming signals on the port. *Return type:* float
`source_latency_early_fall_max`	Returns the maximum early fall source insertion delay specified by an explicit `set_clock_latency` at the port. *Return type:* float
`source_latency_early_fall_min`	Returns the minimum early fall source insertion delay specified by an explicit `set_clock_latency` at the port. *Return type:* float
`source_latency_early_rise_max`	Returns the maximum early rise source insertion delay specified by an explicit `set_clock_latency` at the port. *Return type:* float
`source_latency_early_rise_min`	Returns the minimum early rise source insertion delay specified by an explicit `set_clock_latency` at the port. *Return type:* float
`source_latency_late_fall_max`	Returns the maximum late fall source insertion delay specified by an explicit `set_clock_latency` at the port. *Return type:* float
`source_latency_late_fall_min`	Returns the minimum late fall source insertion delay specified by an explicit `set_clock_latency` at the port. *Return type:* float
`source_latency_late_rise_max`	Returns the maximum late rise source insertion delay specified by an explicit `set_clock_latency` at the port. *Return type:* float
`source_latency_late_rise_min`	Returns the minimum late fall source insertion delay specified by an explicit `set_clock_latency` at the port. *Return type:* float
`user_constant_value`	Returns constant values from netlist or constraints. *Return type*: integer
`x_coordinate`	Returns the x coordinates of the placed port. *Return type:* float
`y_coordinate`	Returns the y coordinates of the placed port. *Return type:* float

Timing Arc Properties

`aocv_derate_capture_clock_early_fall`	Returns graph based AOCV derate factors in AOCV mode for a given timing arc that is a part of early capture clock paths with fall sink pin transitions. *Return type:* float
`aocv_derate_capture_clock_early_rise`	Returns graph based AOCV derate factors in AOCV mode for a given timing arc that is a part of early capture clock paths with rise sink pin transitions. *Return type:* float
`aocv_derate_capture_clock_late_fall`	Returns graph based AOCV derate factors in AOCV mode for a given timing arc that is a part of late capture clock paths with fall sink pin transitions. *Return type:* float
`aocv_derate_capture_clock_late_rise`	Returns graph based AOCV derate factors in AOCV mode for a given timing arc that is a part of late capture clock paths with rise sink pin transitions. *Return type:* float
`aocv_derate_data_early_fall`	Returns AOCV derate values for a timing arc on an early data path with fall transitions. *Return type:* float
`aocv_derate_data_early_rise`	Returns AOCV derate values for a timing arc on an early data path with rise transitions. *Return type:* float
`aocv_derate_data_late_fall`	Returns AOCV derate values for a timing arc on a late data path with fall transitions. *Return type:* float
`aocv_derate_data_late_rise`	Returns AOCV derate values for a timing arc on a late data path with rise transitions. *Return type:* float
`aocv_derate_launch_clock_early_fall`	Returns graph based AOCV derate factors in AOCV mode for a given timing arc that is a part of early launch clock paths with fall sink pin transitions. *Return type:* float

`aocv_derate_launch_clock_early_rise`	Returns graph based AOCV derate factors in AOCV mode for a given timing arc that is a part of early launch clock paths with rise sink pin transitions. *Return type:* float
`aocv_derate_launch_clock_late_fall`	Returns graph based AOCV derate factors in AOCV mode for a given timing arc that is a part of late launch clock paths with fall sink pin transitions. *Return type:* float
`aocv_derate_launch_clock_late_rise`	Returns graph based AOCV derate factors in AOCV mode for a given timing arc that is a part of late launch clock paths with rise sink pin transitions. *Return type:* float
`aocv_stage_count_capture_clock_early`	Returns graph based AOCV stage count values in AOCV mode for a given timing arc that is a part of early capture clock paths. *Return type:* float
`aocv_stage_count_capture_clock_late`	Returns graph based AOCV stage count values in AOCV mode for a given timing arc that is a part of late capture clock paths. *Return type:* float
`aocv_stage_count_data_early`	Returns AOCV stage count values for a timing arc on an early data path. *Return type:* float
`aocv_stage_count_data_late`	Returns AOCV stage count values for a timing arc on a late data path. *Return type:* float
`aocv_stage_count_launch_clock_early`	Returns AOCV stage count values in AOCV mode for a given timing arc that is a part of early launch clock paths. *Return type:* float
`aocv_stage_count_launch_clock_late`	Returns AOCV stage count values in AOCV mode for a given timing arc that is a part of late launch clock paths. *Return type:* float
`arc_type`	Returns the Liberty `timing_type` attribute of the arc. *Return type:* string
`delay_max_fall`	Returns the fall delay for the max corner. *Return type:* float
`delay_max_rise`	Returns the rise delay for the max corner. *Return type:* float

`delay_mean_max_fall`	Returns the mean component of the maximum falling delay value for the timing arc. *Return type*: float
`delay_mean_max_rise`	Returns the mean component of the maximum rising delay value for the timing arc. *Return type*: float
`delay_mean_min_fall`	Returns the mean component of the minimum falling delay value for the timing arc. *Return type*: float
`delay_mean_min_rise`	Returns the mean component of the minimum rising delay value for the timing arc. *Return type*: float
`delay_min_fall`	Returns the fall delay for the min corner. *Return type:* float
`delay_min_rise`	Returns the rise delay for the min corner. *Return type:* float
`delay_sigma_max_fall`	Returns the 1-sigma variation component of the maximum falling delay for the specified timing arc. *Return type*: float
`delay_sigma_max_rise`	Returns the 1-sigma variation component of the maximum rising delay for the specified timing arc. *Return type*: float
`delay_sigma_min_fall`	Returns the 1-sigma variation component of the minimum falling delay for the specified timing arc. *Return type*: float
`delay_sigma_min_rise`	Returns the 1-sigma variation component of the minimum rising delay for the specified timing arc. Return type: float
`delta_delay_max_fall`	Returns the delta fall delay for the max corner. *Return type:* float
`delta_delay_max_rise`	Returns the delta rise delay for the max corner. *Return type:* float
`delta_delay_min_fall`	Returns the delta fall delay for the min corner. *Return type:* float

`delta_delay_min_rise`	Returns the delta rise delay for the min corner. *Return type:* float
`is_cellarc`	Returns a value of `true` if the arc belongs to a cell. *Return type:* boolean
`is_disabled`	Returns a value of true if a `set_disable_timing` has been applied on the supplied library timing arc. *Return type:* boolean
`mode`	Returns the mode specified for the timing arc in the library. *Return type:* string
`object_type`	Returns the object type `timing_arc`. *Return type:* string
`sdf_cond`	Returns the `sdf_cond` expression specified for the timing arc in the library. *Return type:* string
`sdf_cond_end`	Returns the `sdf_cond_end` expression specified for the timing arc in the library. *Return type:* string
`sdf_cond_start`	Returns the `sdf_cond_start` expression specified for the timing arc in the library. *Return type:* string
`sense`	Returns the sense of the timing arc. *Return type:* string
`sink_pin`	Returns the sink pin of the delay arc, or the reference pin of the check arc. *Return type:* collection
`source_pin`	Returns the source pin of the delay arc, or the signal pin of the check arc. *Return type:* collection
`when`	Returns the `when` condition specified for the timing arc in the library. *Return type:* string

Timing Path Properties

`arrival`	Returns the arrival time at the timing path. *Return type*: float
`arrival_mean`	Returns the mean component of the arrival time for the timing path. *Return type*: float
`arrival_sigma`	Returns the sigma component of the arrival time for the timing path. *Return type*: float
`capture_clock_path`	Returns the capturing clock path of the timing path. This property can be used only for `path_type full_clock`. *Return type:* collection
`capturing_clock`	Returns the capturing clock of the timing path. *Return type:* collection
`capturing_clock_close_edge_time`	Returns the closing edge time of the capturing clock. *Return type:* float
`capturing_clock_close_edge_type`	Returns the capturing clock edge direction: `rise` or `fall`. *Return type:* string
`capturing_clock_is_inverted`	Returns a value of true if the phase of the clock at the end point of the capture clock path is opposite to the phase of the capturing clock at the source. *Return type:* boolean
`capturing_clock_is_propagated`	Returns a value of `true` if the capturing clock is in the propagated mode, at the clock end point. *Return type:* boolean
`capturing_clock_latency`	Returns the ideal or propagated latency of the capturing clock. *Return type:* float
`capturing_clock_open_edge_type`	Returns the opening edge time of the capturing clock. *Return type:* string
`capturing_clock_pin`	Returns the name of the capturing clock pin. *Return type:* collection

`capturing_clock_source_arrival_time`	Returns the capturing clock arrival time at the clock root pin/port of the capturing clock path. This value is a sum of the clock edge time and the source latency constraints specified at the clock root. **Note:** You must specify `report_timing -path_type full_clock` in order to return capturing clock arrival time information. *Return type:* float
`capturing_clock_source_arrival_time_mean`	Returns the mean component of the capturing clock arrival time at the clock root pin/port of the capturing clock path. *Return type:* float
`capturing_clock_source_arrival_time_sigma`	Returns the sigma component of the capturing clock arrival time at the clock root pin/port of the capturing clock path. *Return type:* float
`capturing_point`	Returns a pointer to the end point of the timing path. *Return type:* collection
`capturing_point_is_level_sensitive`	Returns a value of `true` if the end point of the data path is level sensitive. *Return type:* boolean
`check_delay`	Returns the check delay value for the timing path. *Return type:* float
`check_delay_mean`	Returns the mean component of the check delay value for the timing path. *Return type:* float
`check_delay_sigma`	Returns the sigma component of the check delay value for the timing path. *Return type:* float
`clock_path_end_point`	Returns the last point in the capture clock path of the timing path. *Return type:* collection
`clock_uncertainty`	Returns the clock uncertainty of the timing path. *Return type:* float
`cppr_adjustment`	Returns the CPPR adjustment of the timing path. *Return type:* float

`cppr_adjustment_mean`	Returns the mean component of the CPPR adjustment of a timing path. *Return type:* float
`cppr_adjustment_sigma`	Returns the sigma component of the CPPR adjustment of a timing path. *Return type:* float
`cppr_branch_point`	Returns the CPPR branch point of the given timing path. *Return Type*: collection
`drive_adjustment`	Returns the drive adjustment value for the timing path. Return type: float
`drive_adjustment_mean`	Returns the mean component of the drive adjustment value for the timing path. Return type: float
`drive_adjustment_sigma`	Returns the sigma component of the drive adjustment value for the timing path. Return type: float
`external_delay`	Returns the value of the `set_output_delay` constraint at the end point of the path. *Return type:* float
`gba_slack`	Returns the original slack time for the timing path – before PBA retiming is performed. *Return type:* float
`gba_slack_mean`	Returns the mean component of the slack time for the timing path – before PBA retiming is performed. *Return type:* float
`gba_slack_sigma`	Returns the sigma component of the slack time for the timing path – before PBA retiming is performed. *Return type:* float
`hold`	Returns the hold time at the end point of the path. *Return type:* float
`is_clock_gating_hold`	Returns a value of `true` if the path is a clock gating hold check. *Return type*: boolean
`is_clock_gating_setup`	Returns a value of `true` if the path is a clock gating setup check. *Return type*: boolean

`is_transparent_latch`	Returns a value of `true` if the path is a transparent latch path. *Return type:* boolean
`launch_clock_path`	Returns the launching clock path of the timing path. This property can be used only for path_type full_clock. *Return type:* collection
`launching_clock`	Returns the launching clock of the timing path. *Return type:* collection
`launching_clock_is_inverted`	Returns a value of true if the phase of the clock at the start point of the launch clock path is opposite to the phase of the launch clock at the source. *Return type:* boolean
`launching_clock_is_propagated`	Returns a value of `true` if the launching clock is in the propagated mode, at the clock end point. *Return type:* boolean
`launching_clock_latency`	Returns the ideal or propagated latency of the launching clock. *Return type:* float
`launching_clock_latency_mean`	Returns mean component of latency of the launching clock. *Return type:* float
`launching_clock_latency_sigma`	Returns sigma component of latency of the launching clock. *Return type:* float
`launching_clock_open_edge_time`	Returns the opening edge time of the launching clock. *Return type:* float
`launching_clock_open_edge_type`	Returns the launching clock's open edge type at the launching point of the timing path. In case of input delay path with respect to a clock it will return the clock edge type, which is triggering the path. *Return type:* string
`launching_clock_source_arrival_time`	Returns the launching clock arrival time at the clock route pin/port of the launching clock path. This value is the sum of the clock edge time plus any source latency constraint specified at the clock root. **Note:** You must specify `report_timing -path_type full_clock` in order to return launching clock arrival time information. *Return type:* float
`launching_clock_source_arrival_time_mean`	Returns the mean component of the launching clock arrival time at the clock route pin/port of the launching clock path.

`launching_clock_source_arrival_time_sigma`	Returns the sigma component of the launching clock arrival time at the clock route pin/port of the launching clock path. *Return type:* float
`launching_input_delay`	Returns the input delay if the timing path starts from an input or output port. This value is set using `set_input_delay`. *Return type:* float
`launching_point`	Returns a pointer to the start point of the timing path. *Return type:* collection
`launching_point_is_level_sensitive`	Returns a value of `true` if the begin point of the data path is level sensitive. *Return type:* boolean
`object_type`	Returns the object type `timing_path`. *Return type:* string
`other_end_arrival`	Returns the capture clock path's begin time, In case of SOCV analysis, mean+N*sigma quantile value is reported. *Return type:* float
`other_end_arrival_mean`	Returns the mean component of capture clock path's begin time. *Return type:* float
`other_end_arrival_sigma`	Returns the sigma component of capture clock path's begin time. *Return type:* float
`path_delay`	Returns the delay of data path component of a timing path. In case of SOCV analysis, mean+N*sigma quantile delay is reported. *Return type:* float
`path_delay_mean`	Returns the mean delay component of data path of a timing path. *Return type:* float
`path_delay_sigma`	Returns the sigma delay component of data path of a timing path. *Return type:* float
`path_group`	Returns a collection of path groups associated with the timing path. *Return type:* collection
`path_group_name`	Returns the name of the path group for the timing path. *Return type:* string

`path_type`	Returns a value of `max` if the path is for setup, or `min` if the path is for hold. *Return type:* string
`period`	Returns the time duration between the launching clock edge and capturing clock edge of the path. The launching clock edge indicates the launching clock time when the signal at the start point of the path is launched, and the capturing clock edge indicates the capturing clock time when the signal at the path end point is captured. For both of these edges, their ideal clock cycle times are considered for calculating the path period. For those paths, where either the launching or capturing clock is absent or asynchronous, the property prints NA. The property works for both setup (late) paths, and hold (early) paths. Usually the path period is decided by a simple subtraction of effective launching edge from the effective capturing edge of the path. Under certain scenario, these edges are moved back or ahead in time, and accordingly the path period is also affected. The following factors impact period: a) Multicycle paths: The launch or capture edge will move by the cycle factor depending on the type of multi cycle exception, and this cycle factor is taken into account while computing the period. b) Max delay paths: Any max_delay specification overrides the relationship between capturing and launching edges and sets it to the max_delay value. The period of the path is thus returned as equal to the specified max delay value for sequential max delay paths. Note that a combinational max delay path has at least one of the launch or capture clocks either absent or asynchronous, and hence the effective period is not calculated for them. c) Clock uncertainty: The uncertainties applicable to the interaction of launch and capture clocks for this particular path also affect the effective edges, but this factor is not taken into consideration while computing the period. d) Unconstrained paths: The unconstrained paths have no capture clock and hence period can not be calculated for them. The property will return NA. *Return value*: float For more information, see "Examples".
`recovery`	Returns the recovery time at the end point of the path. *Return type:* float

`removal`	Returns the removal time at the end point of the path. *Return type:* float
`required_time`	Returns the required time for the timing path. *Return type:* float
`required_time_mean`	Returns the mean component of the required time for the timing path. *Return type:* float
`required_time_sigma`	Returns the sigma component of the required time for the timing path. *Return type:* float
`setup`	Returns the setup time at the end point of the path. *Return type:* float
`skew`	Returns the latency difference between the capturing clock latency and launching clock latency of the path (after considering the effect of CPPR factor). *Return type:* float
`slack`	Returns the slack time for the timing path. *Return type:* float
`slack_mean`	Returns the mean component of the slack time for the timing path. *Return type:* float
`slack_sigma`	Returns the sigma component of the slack time for the timing path. *Return type:* float
`slack_stddev`	Returns standard deviation of the slack. *Return type:* float
`spatial_distance`	Reports the bounding box distance for spatial derating applied on a timing path. *Return type:* float
`time_borrowed`	Returns the amount of time borrowed from the timing end point. *Return type:* float
`time_lent`	Returns the amount of time lent to the timing start point. *Return type:* float

`timing_points`	Returns a collection of the timing points that make up the path. *Return type:* collection
`underated_slack`	Returns the underated slack time for the timing path. The underated slack value is available with timing path object only if it is generated using the `-derate` parameter. *Return type:* float
`view_name`	Returns the analysis view name associated with the timing path. *Return type*: string

Timing Point Properties

`arrival`	Returns the arrival time at the timing point. *Return type*: float
`arrival_mean`	Returns the mean component of the arrival time at the timing point. *Return type*: float
`arrival_sigma`	Returns the sigma component of arrival time at the timing point. *Return type*: float
`delay`	Returns the delay value for the timing point. *Return type:* float
`delay_mean`	Returns the mean component of the delay value for the timing point. *Return type:* float
`delay_sigma`	Returns the sigma component of the delay value for the timing point. *Return type:* float
`fanin_arc`	Returns the delay arc for the corresponding timing point. *Return type*: collection
`object_type`	Returns the `timing_point` object type. *Return type:* string
`pin`	Returns a collection of pins or ports of the timing point. *Return type:* collection

`si_object`	Returns the `si_victim` object type that can be used to query SI delay attributes. **Note**: To enable SI objects, you can set the `timing_report_enable_si_debug` global variable to `true`.
`slew`	Returns the path-specific slew of the timing point. *Return type:* float
`slew_mean`	Returns the mean component of the path-specific slew of the timing point. *Return type:* float
`slew_sigma`	Returns the sigma component of the path-specific slew of the timing point. *Return type:* float
`transition_type`	Returns the transition type of the timing point - rise or fall. *Return type:* string

Signal Integrity Attributes

The signal integrity attributes can be queried using the following object types:

- `si_victim`
- `si_attacker`
- `noise_victim`: Returns a collection of `si_victim` objects that can be used to query glitch attributes.
  Return type: collection

These are explained below.

si_victim Properties

The following information is saved under the `si_victim` object type:

		Valid for Properties		
Property Name	**Description**	**si_victim**	**si_object**	**noise_victim**
`annotated_glitch`	Reports the annotated glitch. *Return type:* float	N	N	Y

`attackers`	Returns a collection of `si_attacker` objects that can be used to query attacker attributes. *Return type:* collection	Y	Y	Y
`capacitance`	Returns the total capacitance of victim net. *Return type:* float	Y	Y	Y
`coupling_capacitance`	Returns the total coupling capacitance of a victim net. *Return type:* float	Y	Y	Y
`glitch_area`	Returns the glitch area. *Return type:* float	N	N	Y
`glitch_width`	Returns the glitch width. *Return type:* float	N	N	Y
`hierarchical_name`	Returns the hierarchical name of a victim net. *Return type:* string	Y	Y	Y
`input_peak`	Returns the Receiver Input Peak (RIP) value. *Return type:* float	N	N	Y
`input_failure_threshold`	Returns the failure threshold at RIP. This will always be reported irrespective of whether failure happens at RIP or ROP. *Return type:* float	N	N	Y
`level`	Returns the noise level - VL or VH. *Return type:* string	N	N	Y
`num_active_attackers`	Returns the number of active attackers. *Return type:* float	Y	Y	Y

`num_attackers`	Returns the number of total attackers (distinct + accumulated small attacker - as 1). The attackers that are dropped from the accumulated small attackers will not be included with the total attackers. *Return type:* float	Y	Y	Y
`object_type`	Returns the object type `si_victim`. *Return type:* string	Y	Y	Y
`prop_glitch`	Reports the propagated glitch seen at the victim driver output. *Return type:* float	N	N	Y
`receiver_peak`	Returns the Receiver Output Peak (ROP) value. *Return type:* float	N	N	Y
`receiver_peak_threshold`	Returns the failure threshold at ROP. This will always be reported irrespective of whether failure happens at RIP or ROP. *Return type:* float	N	N	Y
`resistance`	Returns the total resistance of a victim net. *Return type:* string	Y	Y	Y
`transition`	Returns the victim transition direction (rise or fall). *Return type:* string	Y	Y	N
`view_type`	Returns the type of analysis (early or late). *Return type:* string	Y	Y	N

si_attacker Properties

The following Information is saved under the `si_attacker` object type for each attacker of the victim net in the path:

Property Name (for each attacker retrieved from a victim)	Description
`coupling_capacitance`	Returns the coupling of a victim net. *Return type:* float
`hierarchical_name`	Returns the hierarchical name of an attacker. *Return type:* string
`is_active`	Returns whether an attacker is active or not. *Return type:* string
`noise_peak`	Returns the Xtalk delay bump induced by an attacker. *Return type:* float
`object_type`	Returns the `si_attacker` object type. *Return type:* string
`slew`	Returns the slew of attackers. *Return type:* float
`state`	Returns the status of attackers. The status descriptions are given below: • SY: Synchronous • INF: Attacker with infinite timing window • PE: Filtered due to physical exclusion • UF: Filtered by user • CN: Filtered because it is a constant net • SB: Filtered due to small bump • TA: Filtered because of timing window does not overlap • LC: Filtered due to logical correlation • NA: Attacker status is not known *Return type:* string

Examples

The following `get_property` example shows the usage of SI attributes to query noise information:

```
source pinlist.tcl
set chan [open attribute.log w]
foreach pin $pinList {
   set vicArray [get_property [get_pins $pin] noise_victim]
   foreach_in_collection vic $vicArray {
       set input_peak [ get_property $vic input_peak]
       #if { $input_peak > 250 } {}

       set vic_name [ get_property $vic hierarchical_name ]
       set x [ get_property [get_net -of_objects $vic_name ] name ]
       set level [ get_property $vic level]
       set ccap [ get_property $vic capacitance]
       set vdd [get_property [get_pins $vic_name] power_rail_voltage_max]
       set libs [get_libs -of_objects [get_lib_cells -of_objects [get_cells -of_objects [get_pins
$vic_name]]]]
       foreach_in_collection libName $libs {set library [get_property $libName hierarchical_name]}
       set prop_glitch [ get_property $vic prop_glitch]
       set cap [format %.2f $ccap]
       set annotated_glitch [ get_property $vic annotated_glitch]
       set receiver_peak [ get_property $vic receiver_peak]
       set input_peak_threshold [ get_property $vic input_failure_threshold ]
       set receiver_peak_threshold [ get_property $vic receiver_peak_threshold]
       puts $chan "-------------------------------------------------------------------------------"
         puts $chan "Peak(mV) Level TotalCap(fF) Vdd(V) Library VictimNet"
         puts $chan "$input_peak $level $cap $vdd $library $vic_name {$x}"

         if { $receiver_peak > 0 } {
           puts $chan "Receiver output peak:"
           puts $chan "Value ReceiverNet"
           puts $chan "$receiver_peak ( $receiver_peak_threshold) $vic_name (CELL)"
         }

         if { $input_peak > 0 } {
           puts $chan "Receiver Input peak:"
           puts $chan "Value ReceiverNet"
           puts $chan "$input_peak ( $input_peak_threshold) $vic_name (CELL)"
         }

         puts $chan "\nConstituents:"
         puts $chan "Source Peak(mV) Offset(ps) Slew(ps) Xcap(fF) Vdd(V) Edge Status Net"
         set aggs [ get_property $vic attackers ]
         foreach_in_collection current_agg $aggs {

           set hierarchical_name [ get_property $current_agg hierarchical_name ]
           set noise_peak [ get_property $current_agg noise_peak]
           set state [ get_property $current_agg state]
           set slew [ get_property $current_agg slew]
           set edge [ get_property $current_agg transition]
```

```
            if {[string match ~* $hierarchical_name]} {
            set vdd [get_property [get_pins $vic_name] power_rail_voltage_max]
            } else {
            set vdd [get_property [get_pins -of_objects [get_nets $hierarchical_name]]
power_rail_voltage_max]
            }
            set coupling_capacitance [ get_property $current_agg coupling_capacitance]

            puts $chan "Cpl: $noise_peak $slew $edge $vdd $coupling_capacitance $state $hierarchical_name"
            }
          if { $prop_glitch > 0 } {
            puts $chan "Prp: $prop_glitch"
           }


}
close $chan
```

User-Defined Properties

The user-defined properties allow you to define certain properties which can be interpreted by the software in a predefined way. When these properties are not specified, the software will revert to the default behavior, as applicable.

Property Name	Type	Description
`capacitive_load_unit`	lib	Provides the load unit mentioned in the library.
`time_unit`	lib	Provides the time unit mentioned in the library.
`aocv_weight`	lib_cell	Defines AOCV weight for a given cell.
`k_input_switching_window_fall*`	lib	Defines the switching window threshold for fall signal during the application of SSI (simultaneous switching input) derating feature. *Default*: 1
`k_input_switching_window_rise*`	lib	Defines the switching window threshold for rise signal during the application of SSI (simultaneous switching input) derating feature. *Default*: 1

* When the above properties are defined in SSI derating environment, the delay of the respective cell multiplied by the above factor will be compared with the difference in (rise/fall) signal arrivals at two inputs to determine if switching derate needs to be applied to cell delay.

Examples

- The following command returns the hierarchical name for the pin contained in the collection referenced by `$pin`:

  ```
  get_property $pin hierarchical_name
  ```

- The following example shows how you can query a user-defined property from `stage_weighting` library at the cell level:

```
set_property [get_lib_arcs -of fast/BUFX2] stage_weighting 3 // sets the value
get_property [get_lib_arcs -of fast/BUFX2] stage_weighting
```

- The following example shows power and ground pins for the corresponding library pins:

```
get_property [get_lib_pins CELLA/A] related_power_pin
get_property [get_lib_pins CELLA/A] related_ground_pin
```

- The following example shows active clocks:

```
get_property [get_clocks Iclk1] is_active true
```

- The following commands define the switching window threshold for rise signal:

```
define_property -type float -object_type lib k_input_switching_window_rise
set_property [get_libs slow] k_input_switching_window_rise 1.1
```

- The following example shows that the phase shift is 2.0 ns, as returned by the "period" property:

```
report_timing -clock_to CLK_W_3 -clock_from CLK_W_3
```

```
  Path 1: MET Setup Check with Pin seg3/u14/CK
  Endpoint: seg3/u14/D (^) checked with leading edge of 'CLK_W_3'
  Beginpoint: seg3/u9/Q (^) triggered by leading edge of 'CLK_W_3'
  Path Groups: {CLK_W_3}
  Other End Arrival Time 0.621
  - Setup 0.186
  + Phase Shift 2.000
  = Required Time 2.434
  - Arrival Time 2.062
  = Slack Time 0.372
  Clock Rise Edge 0.000
  + Clock Network Latency (Prop) 0.682
  = Beginpoint Arrival Time 0.682
  ……..
  > set path [rt -clock_to CLK_W_3 -clock_from CLK_W_3 -collection]
  0x14
  > get_property $path period
  2.000
```

- In this following example, there is a clock reference for launch clock (Null phase- @), so the software does not show any clock period for this path:

```
report_timing -clock_to CLK_W_3 -early
```

```
  Path 1: VIOLATED Hold Check with Pin seg3/u3/CK
  Endpoint: seg3/u3/D (v) checked with leading edge of 'CLK_W_3'
  Beginpoint: IN3 (v) triggered by leading edge of '@'
  Path Groups: {CLK_W_3}
  Other End Arrival Time 0.682
  + Hold -0.034
```

```
+ Phase Shift 0.000
= Required Time 0.648
Arrival Time 0.117
Slack Time -0.531
Clock Rise Edge 0.000
+ Input Delay 0.000
+ Drive Adjustment -0.014
= Beginpoint Arrival Time -0.014
-----------------------------------------------------------------------------------------
Load  Slew   Delay   Incr    Arc        Fanout  Cell  Arc           User   Arrival Instance
                     Delay   Annotation                           Derate Time
-----------------------------------------------------------------------------------------
0.013 0.029                  WLM        1       IN3     v                -0.014
0.013 0.057  0.054   0.000   WLM        1       CLKBUF  A v -> Y v 1.000  0.040   seg3/u1
0.003 0.050  0.077   0.000   WLM        1       CLKBUF  A v -> Y v 1.000  0.117   seg3/u2
0.003 0.050  0.000   0.000   WLM                DFF     D v        1.000  0.117   seg3/u3
-----------------------------------------------------------------------------------------

> set path [report_timing -clock_to CLK_W_3 -collection -early]
0x16
> get_property $path period
NA
```

Related Commands

- report_property
- set_property
- define_property
- report_object_property_names

index_collection

`index_collection` *`base_collection index`*

Returns a collection containing the object that exists at the specified index of the specified object collection.

Parameters

`base_collection`	Specifies the base collection.
`index`	Specifies the index number. You can specify a number from `0` to *`collection_size-1`*.

Examples

- The following command returns the object that exists at the index number `2` in the object collection referenced by `$nets`:
  index_collection $nets 2

query_objects

`query_objects` *`collection`* `[-limit` *`value`*`]`

Displays the names of the objects contained inside the specified collection.

⚠ `query_objects` displays object names only. To return a tcl list of object names, the `get_object_name` command may be used instead.

Parameters

`collection`	Specifies the object collection for which to print names.
`-limit` *`value`*	Specifies the maximum number of object names to print. Specify a value of `0` to query the names of all of the objects in the collection. *Default*: Prints the first 100 object names in the collection.

Examples

- The following commands create collections of pins and nets, add them together to create a new collection, and prints the names of the objects in the collection:

```
set pins [get_pins {i_block1/b1_Sin1 i_block1/b1_CLK1 i_block1/b1_Sout1}]
set nets [get_nets {Sin1 CLK1 Sout1}]


add_to_collection newVar $pins
add_to_collection newVar $cells


query_objects $newVar
```

  The software returns the following information:

```
i_block1/b1_Sin1 i_block1/b1_CLK1 i_block1/b1_Sout1 Sin1 CLK1 Sout1
```

Related Information

- `get_object_name`

range_collection

```
range_collection
[-help]
base_collection
from
to
```

Returns a sub-collection of the base collection. The sub-collection range lies between the ‘from’ and ‘to’ (including ‘from’ and ‘to’) index numbers specification of this command.

Parameters

`-help`	Prints out the command usage.
`base_collection`	Specifies the base collection.
`from`	Specifies the ‘from’ index number.
`to`	Specifies the ‘to’ index number.

The index number can be an integer or an expression, such as, “end OR end-integer” (with no spaces within the expression).

If the specified to/from value is negative (less than 0), then the software will change this value to 0. If the value is greater than the “collection_size – 1”, then it will be changed to “collection_size – 1”.

Examples

- The following example sets `$points` as the base collection:

```
> set points [get_cells *]
0x13
> sizeof_collection $points
22
```

- The following example returns the sub-collection of the base collection `$points` between the index number range from 2 to 4:

```
> range_collection $points 2 4
clk_buf2 clk_buf3 clk_buf4
0x16
> range_collection $points 4 2
clk_buf2 clk_buf3 clk_buf4
0x17
```

- The following example returns a sub-collection starting from index number 0 till the end of the base collection:

```
> range_collection $points 0 end
clk_buf1 pad1 clk_buf2 clk_buf3 clk_buf4 buf1 buf2 buf3 buf4 mem dff1 dff2 clkand2 pu_cell_1
```

```
pu_lat_1 u1 cg4 genLoop comboLoop pad2 dff3 latch1
0x18
```

- The following example returns a sub-collection starting from end-integers 4 to 2:

```
> range_collection $points end-4 end-2
genLoop comboLoop pad2
0x19
```

- The following example shows a range starting from a negative value to a value greater than the collection size. The following warning messages are displayed:

```
> range_collection $points -5 100
**WARN: (TCLCMD-1419): 'from' index with value -5 is less than start index, so it is changed to 0 .
**WARN: (TCLCMD-1419): 'to' index with value 100 is greater than end index, so it is changed to 21 .
clk_buf1 pad1 clk_buf2 clk_buf3 clk_buf4 buf1 buf2 buf3 buf4 mem dff1 dff2 clkand2 pu_cell_1
pu_lat_1 u1 cg4 genLoop comboLoop pad2 dff3 latch1
0x1a
```

- The following example shows the warning message displayed for a negative start index number:

```
> range_collection $points end-25 end-20
**WARN: (TCLCMD-1419): 'from' index with value -4 is less than start index, so it is changed to 0 .
clk_buf1 pad1
0x1b
```

- The following command shows an error if the specified index number is not an integer:

```
> range_collection $points s end
**ERROR: (TCLCMD-1418): Bad index 's': must be integer? or end?[-]integer?
```

Related Commands

- index_collection

remove_from_collection

```
remove_from_collection
base_collection
[-intersect]
collection_or_obj_list
```

Creates a new collection by removing the objects specified in one collection from another collection.

Parameters

base_collection	Specifies the base collection of objects. Objects from the second collection or object list are removed from this collection to create the new collection.
-intersect	Provides intersection of base collection elements with second collection of objects or lists.
collection_or_obj_list	Specifies the collection or list of objects that should be removed from the base collection to create the new collection.

Examples

- The following commands create a collection of pins that does not include clock pins:

  ```
  set allP [get_pins *]
  set all_pin_no_clock [remove_from_collection $allP [get_pins -filter "is_clock == true"]]
  ```

  You can use the `sizeof_collection` command to see that the `all_pin_no_clock` collection contains fewer objects (that is, no clock pins) than the original `allP` collection:

  ```
  sizeof_collection $allP
  50
  sizeof_collection $all_pin_no_clock
  47
  ```

Related Information

- add_to_collection
- append_to_collection
- compare_collections

report_object_property_names

```
report_object_property_names [-help]
[-obj_type {cell clock design lib lib_cell lib_pin lib_timing_arc net path_group pin port timing_arc
timing_path timing_point}]
```

Lists all of the properties associated with the specified object type.

If you do not specify any object type, this command lists all the supported object types, and all the properties associated with those object types.

Parameters

`-help`	Prints out the command usage.
`-obj_type`	Lists the properties associated with the specified object type. The following object types are supported: cell, clock, design, lib, lib_cell, lib_pg_pin, lib_pin, lib_timing_arc, net, path_group, pg_net, pg_pin, pin, port, si_attacker, si_victim, timing_arc, timing_path, or timing_point Additional SI properties can be included with the `timing_point` object. To enable SI properties, you can set `timing_report_enable_si_debug` global variable to `true`.

Related Commands

- `define_property`
- `get_property`
- `report_property`
- `reset_property`
- `set_property`

report_property

```
report_property
[-help]
{collection | list_of_collections}
[-property_list list of properties]
[{> | >>} filename]
```

Reports the properties associated with each object in the specified collection.

For properties that return a populated collection, the command output displays `{...}`. For any empty collections,`{}` will be displayed.

Parameters

`-help`	Prints out the command usage.
*collection	list_of_collections*	Specifies a collection, or a list of collections to report. The collections can contain any of the following object types: cell, clock, design, lib, lib_cell, lib_pin, lib_pg_pin, pg_net, pg_pin, lib_timing_arc, net, path_group, pin, port, timing_arc, timing_path, or timing_point
`-property_list` *list of properties*	Specifies a list of valid property names to report. **Note**: If the `-property_list` parameter is not specified, then all the properties associated with a collection of objects will be reported.
{`>` | `>>`} *filename*	When specified with >, the software writes the report to the specified file name. If the file already exists, the software overwrites it. When specified with >>, the software writes the report to the specified file name. If the file already exists, the software concatenates the report to the end of the file. You can add the `.gz` extension to the file name to generate a compressed report. **Note:** The file name must be the last argument in the list.

Examples

- The following command shows the values of the specified properties:

```
report_property [get_pins TU11/A] -property_list {name direction clocks}

property | value
name | TU11/A
direction | in
clocks | {...}
```

Related Commands

- get_property
- define_property
- report_object_property_names
- set_property
- reset_property

sizeof_collection

`sizeof_collection` *`collection`*

Returns the total number of objects contained in the specified collection. If the collection contains different types of objects, `sizeof_collection` returns the sum of all the objects of different types in the collection.

Parameters

`collection`	Specifies the collection.

sort_collection

```
sort_collection
[-help]
collection
[-dictionary]
{property_or_list_of_properties}
[-descending]
```

Returns a sorted collection of objects based on the property.

Parameters

`-help`	Prints out the command usage.
collection	Specifies the collection to sort. The collection must be a homogenous collection of any one of the following object types: `cell`, `clock`, `design`, `lib`, `lib_cell`, `lib_pin`, `lib_timing_arc`, `net`, `path_group`, `pin`, `port`, `timing_arc`, `timing_path`, or `timing_point`.
`-descending`	Sorts the objects in the collection by descending order. *Default:* Sorts the collection in ascending order.
`-dictionary`	Sorts the objects in a collection by dictionary order.
property | *list_of_properties*	Sorts the collection by the specified property. You can specify any property associated with the object type. You can specify one or more properties. If you specify more than one property, the software sorts the collection in the order by which you specified the properties.

Examples

- The following command returns a collection of all the input ports in the design sorted on the basis of their hierarchical names:

```
> sort_collection [all_inputs] hierarchical_name

CLK1 CLK2 CLK3 CLK4 IN1 IN2 IN3 IN4
```

2

ART Based Flow Commands

- clear_active_logic_view
- create_active_logic_view
- get_module_view
- set_module_view

clear_active_logic_view

```
clear_active_logic_view
[-help]
```

Resets the reduced timing graph created by `clear_active_logic_view` back to the original timing graph.

Parameters

`-help`	Outputs a brief description of the `clear_active_logic_view` command. For a detailed description of the command, use the `man` command: `man clear_active_logic_view`

create_active_logic_view

```
create_active_logic_view
[help]
[-type {flat_top timing_budget}[-set_dont_touch | -unset_dont_touch ]]
```

Trims the timing graph to ignore logic inside the first level registers in partitions to improve runtime for timing-related commands.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `create_active_logic_view` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man create_active_logic_view`.
`-set_dont_touch`	Sets the dontTouch value as `true`. This freezes I/O interfaces (Interface logics and interface nets) and prevents the opt_design `-post_cts -hold` from optimizing the block interface logics. The ILM ECO data can be loaded back.
`-type {flat_top timing_budget}`	• `flat_top`: Used to mark the top-level timing graph to mask all logic inside the interface logic of each partition. • `timing_budget`: Instructs the command to use the same settings as `create_timing_budget` when marking the top-level timing graph. This option should be specified if you plan to run `time_design` or `opt_virtual` with ART based and later on run `create_timing_budget` based on the resulted timing graph.
`-unset_dont_touch`	Sets dontTouch value as `false`. This resets the dontTouch on I/O logics before reading the ECO data.

Related Topics

- `create_timing_budget`
- `time_design`
- `opt_virtual`
- "Timing Budgeting" chapter in the *Innovus User Guide*
  - Chip-Level Budgets Derived by Using Active Logic View
- "Optimizing Timing" chapter in the *Innovus User Guide*
  - Using Active Logic View for Chip-Level Interface Circuit Timing Closure

get_module_view

```
get_module_view
[-help]
[-top_read_only]
[-hinsts list_of_partition_hinsts]
[-partitions list_of_paritions]
[-tcl_list]
```

Verifies the FlexView mode.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `get_module_view` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man get_module_view`
`-hinsts` *`list_of_partition_hinsts`*	Lists partition hinsts.
`-partitions` *`list_of_paritions`*	Lists partitions.
`-top_read_only`	Specifies if top level is readOnly.
`-tcl_list`	Displays the current settings for the specified parameters in Tcl list format only. If you specify `-tcl_list` without any parameters, the software displays the current settings of all `set_module_view` parameters in Tcl list format.

Examples

- The following command verifies the FlexView mode:

```
get_module_view


{TOP all}
{coreinst/ks_core1/amba_dsp1/ram2p_78kx32 readOnly}
{coreinst/ks_core1/leon1 interface}
```

set_module_view

```
set_module_view
[-help]
[-top_read_only {true | false}]
[-hinsts {list_of_partition_hinsts}]
[-partitions {list_of_paritions}]
[-type {read_only|interface|all}]
```

Specifies the FlexView (`read_only|interface|all`) for the top level and each of the partitions. This way optimization can be limited to the selected sections of the partitions and the top-level logic.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `set_module_view` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man set_module_view`.
`-top_read_only {true | false}`	Sets the top level to the read-only mode. By default, optimization can change the logic at the top level to improve timing. By setting this option to `true`, no change will be done to the entire top-level logic. Note: *Default*: `false`
`-hinsts` *list_of_partition_hinsts*	Applies the selected FlexView mode to the instance name or list of instance names of partitions.
`-partitions` *list_of_partitions*	Applies the selected FlexView mode to a partition or list of partitions. You have a choice to either specify the instance name of the partition or just the partition itself.
`-type {read_only|interface|all}`	Specifies the FlexView mode for the specified partition(s). • `read_only` : Specifies that the optimization will not be performed inside the partition. • `interface`: Specifies that optimization will be restricted only to the interface logic of the partition. • `all` : Specifies that optimization will be done for the entire partition. **Note**: The read_only status of the parent partition is not inherited by the child partition.

3

Basic Database Access Tcl Commands and Attributes

- add_line_end_track
- commit_global_net_rules
- compress_special
- connect_global_net
- convert_dbu_to_um
- convert_um_to_dbu
- create_cell_obs
- create_via
- create_via_definition
- create_virtual_shape_on_pg_wires
- decompress_special
- delete_cell_obs
- delete_floating_nets
- delete_global_net_connections
- get_computed_shapes
- get_edge
- get_obj_in_area
- get_rect
- get_transform_shapes
- report_floating_nets
- report_metal_stack
- report_net_wires
- report_preserves
- report_rc
- report_via_def
- report_wire_path
- safety Category Attributes
- tech Category Attributes
- uniquify

add_line_end_track

```
add_line_end_track
[-help]
[-eol_width float]
[-except_eol_spacing float [-within float]]
-layer string
-offset float
[-track_distance float]
[-track_pitch float]
```

Creates a line-end track.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `add_line_end_track` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man add_line_end_track`
`-eol_width` *float*	Specifies the width of the line-end track. *Data_type*: float, optional
`-except_eol_spacing` *float*	Specifies the spacing of the line-end track. *Data_type*: float, optional
`-layer` *string*	Specifies the layer on which the track is to be created. *Data_type*: string, required
`-offset` *float*	Specifies the offset to the core boundary. *Data_type*: float, required
`-track_distance` *float*	Specifies the distance between two sets of tracks. *Data_type*: float, optional
`-track_pitch` *float*	Specifies the track pitch. *Data_type*: float, optional
`-within` *float*	Specifies the extension in the orthogonal direction to find a neighbor wire. *Data_type*: float, optional

Related Information

Floorplanning the Design chapter in *Innovus User Guide.*

commit_global_net_rules

```
commit_global_net_rules [-help]
```

Applies or restores the global net connectivity rules to the design and creates the necessary connections between instances and these global nets.

This command will reset all the power connections based on the current global net rules, and is only needed if somehow the PG connectivity is inconsistent with them (due to reading inconsistent DEF connections, bugs, etc).

Parameters

`-help`	Prints a brief description of the `commit_global_net_rules` command usage. For a detailed description of the command, use the `man` command: `man commit_global_net_rules`

Related Topics

Power Planning and Routing chapter in the *Innovus User Guide.*

compress_special

```
compress_special
[-help]
[-area {x1 y1 x2 y2}]
[-incremental]
[-layers layer+]
[-nets net+]
```

Compresses Power-Grid (PG) based on the specified area, layer, or net.

Parameters

`-help`	Prints a brief description that includes the type and default information for each `compress_special` parameter. For a detailed description of the command and all its parameters, use the man command: `man compress_special`
`-area {x1 y1 x2 y2}`	Specifies the coordinates for the area from which PG needs to be compressed. The `compress_special` command compresses only those special wires and special vias, which exist within the area specified by this parameter. *Default*: All PG structures are compressed. *Data_type*: rect, optional
`-incremental`	Recompresses the PG after editing the PG wires or vias. This parameter can only be specified when the current design is in compressed mode. *Data_type*: bool, optional
`-layers layer+`	Specifies the layers on which PG compression is needed. This parameter requires either LEF layer names or a routing layer index value (for example, `1` for the first routing layer, `2` for the second routing layer, and so on). *Data_type*: (layer)+, optional
`-nets net+`	Specifies the nets from which PG compression is needed. If the specified net does not have any special route on it, it is ignored. *Data_type*: (net)+, optional

Example

The following command compresses `%s special_wires` and `%s special_vias`:

```
compress_special -area {100 100 120 120} -net VDD -incremental
```

Related Topics

`decompress_special`

connect_global_net

```
  connect_global_net
[-help]
global_net_name
{{-type pg_pin -pin_base_name pinNamePattern |
-type tie_hi [-pin_base_name pinNamePattern] |
-type tie_lo [-pin_base_name pinNamePattern]}
{{-single_inst | -singleInst | -sinst} instName |
[{-inst_base_name | -instBasename} instBasenamePattern]
[{-hinst | -hierInst | -module} hierInstName |
-region x1 y1 x2 y2 |
-power_domain powerDomainName | -all]} |
-type net -net_base_name netBasenamePattern
[{-hinst | -hierInst | -module} hierInstName |
-power_domain powerDomainName | -all]}
[-override]
[-verbose]
[-auto_tie]
[-disconnect]
[-netlist_override]
[-non_hierarchical]
```

Adds a new global net connection to the specified global net.

You can use the `connect_global_net` syntax in three general design situations. The three design situations, and the suggested usage for the `connect_global_net` syntax, appear below:

- Connecting pins in a single instance to a global net. In the example below, *`instName`* represents an instance's complete hierarchical name, without any wildcard characters:

  ```
  connect_global_net
  global_net_name
  {-type pg_pin -pin_base_name pinNamePattern | -type tie_hi [-pin_base_name pinNamePattern]
  | -type tie_lo [-pin_base_name pinNamePattern]}
  -single_inst instName
  [-override]
  [-verbose]
  ```

- Connecting pins in multiple instances, or pins in a region, to a global net:

  ```
  connect_global_net
  global_net_name
  {-type pg_pin -pin_base_name pinNamePattern | -type tie_hi [-pin_base_name pinNamePattern]
  | -type tie_lo [-pin_base_name pinNamePattern]}
  [-inst_base_name instBasenamePattern]
  [-hinst hierInstName | -region x1 y1 x2 y2 | -all]
  [-override]
  [-verbose]
  ```

- Connecting nets to a global net:

  ```
  connect_global_net
  global_net_name
  -type net -net_base_name netBasenamePattern
  [-hinst hierInstName | -all]
  [-override]
  [-verbose]
  ```

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `connect_global_net` parameter. For a detailed description of the command and all its parameters, use the `man` command: `man connect_global_net`
`-all`	Applies the global net connection to all instances in the design. *Data_type*: bool, optional
`-auto_tie`	Support supply/ground sensitivity. In a command for PG-pin_base_name connections, the option can only be specified together with `-type pg_pin`. In addition to connecting power pin VDD to global net VDD for each instance, all the instance tie-high pins whose related power pin (as specified in CPF, .lib, or LEF) is VDD, will automatically be tied to global net VDD. For example, `connect_global_net VDD -type pg_pin -pin_base_name VDD -auto_tie` *Data_type*: bool, optional
`-disconnect`	Disconnect p/g net connection. The option works only with `-sinst`, `-type pg_pin` and `-pin_base_name` options. **Note:** This option is mutually exclusive with `global_net_name` argument. *Data_type*: bool, optional
`global_net_name`	The name of the global net to which the specified pins, modules, or nets connect. *Data_type*: string, optional
`-hinst` *`hierInstName`*	Applies the global net connection to instances under *`hierInstName`*. Only the pins of the instances or the local nets under the specified module or hierarchical instance are connected. This parameter is mutually exclusive with the `-region` parameter. **Note:** You can also use the following spelling variations for this argument: • `-hierInst` • `-module` *Data_type*: string, optional
`-inst_base_name` *`instBasenamePattern`*	
	Specifies the names of leaf instances for which pins are to be connected to the global net. You can use the wildcard (`*`) character to specify a pattern of instance basenames. An instance basename cannot contain the "`/`" character. This parameter is used in conjunction with the `-type pg_pin` parameter, or with the `-type tie_hi` or `-type tie_lo` parameters. **Note:** You can also use the `-instBasename` spelling variations for this argument. *Data_type*: string, optional
`-net_base_name` *`netBaseName`*	Specifies the basenames of nets to connect to the global net. You can use the wildcard (`*`) character to specify a pattern of net basenames. An net basename cannot contain the "`/`" character. This parameter is used in conjunction with the `-type net` parameter. *Data_type*: string, optional
`-netlist_override`	Overrides previous global net connections. The option is similar to the option `-override`. However, in addition to overriding previous global net connections, `-netlist_override` also overrides netlist-specified P/G-pin_base_name connections if applicable, which the option `-override` can't do. *Data_type*: bool, optional
`-non_hierarchical`	Invokes the non hierarchical cell instance. The option is of type boolean and is optional. *Data_type*: bool, optional

`-override`	Specifies that the values that used with the `connect_global_net` command override global net connection values that have been set previously. *Data_type*: bool, optional
`-pin_base_name` *pinNamePattern*	Specifies the pins to connect to the global net. You can use the wildcard (`*`) character to specify a pattern of pin names. This parameter is used in conjunction with the `-type pg_pin` parameter, or optionally with the `-type tie_hi` or `-type tie_lo` parameters. *Data_type*: string, optional
`-power_domain` *powerDomainName*	Applies the global net connection to instances within the specified power domain. *Data_type*: string, optional
`-region` *x1 y1 x2 y2*	Applies the global net connection to instances inside the specified region. The region is specified by the x and y coordinates of the lower left and upper right boundary. This parameter is mutually exclusive with the `-module` parameter. *Data_type*: rect, optional
`-single_inst` *instName*	Connects pins in a single instance to a global net. *-instName* is the instance's complete hierarchical name, without any wildcard characters. **Note:** You can also use the following spelling variations for this argument: • `-sinst` • `-singleInst` *Data_type*: string, optional
`-type {net | pg_pin | tie_lo | tie_hi}`	Specifies the type of connection to be made to the global net. Use one of the following values: • `net`: Specifies that the nets listed with the `-net_base_name` parameter are to be connected. • `pg_pin`: Specifies that the power and ground pins listed with the `-pin_base_name` parameter are to be connected. • `tie_lo`: Specifies that tie low pins be connected. • `tie_hi`: Specifies that tie high pins be connected. *Data_type*: enum, optional
`-verbose`	Specifies that connection statistics and warning messages are displayed in the console. This parameter can be useful during troubleshooting. *Data_type*: bool, optional

Example

The following command connects the vdd! net with the VDD global net:

```
connect_global_net VDD -type net -net_base_name vdd!
```

Related Topics

Power Planning and Routing chapter in the *Innovus User Guide.*

convert_dbu_to_um

```
convert_dbu_to_um
[-help]
[-unit number]
{int | {int_list}}
```

Takes an arbitrary list of ints and returns it in the same form, with each database integer int converted to its user unit floating point equivalent.

Parameters

`-help`	Prints a brief description that includes type and default information for each `convert_dbu_to_um` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man convert_dbu_to_um`
`-unit number`	Specifies the convert factor by which to divide the database integer int. *Value:* One of `100`, `200`, `400`, `800`, `1000`, `2000`, `4000`, `8000`, `10000`, or `20000` *Default:* Uses the LEF `UNITS DATABASE MICRONS` int from the Innovus database technology file, or the OpenAccess equivalent. *Data_type*: enum, optional
`{int | {int_list}}`	
	Specifies the int or int list to convert. You can specify a single int, or an arbitrary list of ints, including points, rectangles and coordinates. *Data_type*: integer or list of integers, required

Examples

- The following command converts the database int `100` into its equivalent user unit int, based on a convert factor of `2000`:
  ```
  convert_dbu_to_um -unit 2000 100
  ```
  The software returns the following information:
  ```
  0.0500
  ```
- The following command converts the database ints for the list of numbers `{1000 1000 2000 2000}` into their equivalent user unit ints, based on a convert factor of `1000`:
  ```
  convert_dbu_to_um -unit 1000 {1000 1000 2000 2000}
  ```
  The software returns the following information for the list of numbers. The list is implied, but not shown.
  ```
  1.000 1.000 2.000 2.000
  ```

Related Topics

`convert_um_to_dbu`

convert_um_to_dbu

```
convert_um_to_dbu
[-help]
[-unit number]
{double | {double_list}}
```

Takes an arbitrary list of doubles and returns it in the same form, with each user unit floating point double converted to its database integer double equivalent.

Parameters

`-help`	Prints a brief description that includes type and default information for each `convert_um_to_dbu` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man convert_um_to_dbu`
`-unit number`	Specifies the convert factor by which to multiply the user unit double. *Value:* One of `100`, `200`, `1000`, `2000`, `10000`, or `20000` Default: Uses the LEF `UNITS DATABASE MICRONS` double from the Innovus database technology file, or the OpenAccess equivalent. *Data_type*: enum, optional
`{double | {double_list}}`	
	Specifies the double or double list to convert. You can specify a single double, or an arbitrary list of doubles, including points, rectangles and coordinates. *Data_type*: float or list of floats, required

Examples

- The following command converts the user unit double `30` into its equivalent database integer double, based on a convert factor of `2000`:
  ```
  convert_um_to_dbu -unit 2000 30
  ```
  The software returns the following information:
  ```
  60000
  ```
- The following command converts the user unit doubles for the list of numbers {`30 40`} into their equivalent database unit doubles, based on a convert factor of `200`:
  ```
  convert_um_to_dbu -unit 200 {30 40}
  ```
  The software returns the following information for the list of numbers. The list is implied, but not shown.
  ```
  6000 8000
  ```

Related Topics

`convert_dbu_to_um`

create_cell_obs

```
create_cell_obs
[-help]
-cell base_cell
[-ignore_pg_net]
-layer layer
[-mask int_value]
{-rects {{llx1 lly1 urx1 ury1} {llx2 lly2 urx2 ury2}...} | -polygon {x1 y1 x2 y2 x3 y3 ...}}
[-spacing value | -design_rule_width value]
```

Helps in adjusting the database. The added obs is saved in the database by its own. You do not need to resource it after restoring the database. It adds an obstruction geometry to a cell on the specified layer and is helpful when you want to add a temporary cell obstruction without changing the cell's LEF and reloading the Innovus.

Parameters

`-help`	Prints a brief description that includes the type and default information for each `create_cell_obs` parameter. For a detailed description of the command and all its parameters, use the man command: `man create_cell_obs`
`-cell base_cell`	Specifies the base cell or ilm cell to be modified. *Data_type*: (base_cell|design|module), required
`-design_rule_width value`	Specifies the design rule width. *Data_type*: float, optional
`-ignore_pg_net`	Ignores the Power/Ground routing while checking for DRC violations (including shorts) involving the added obs. This parameter is equivalent to `LEF MACRO OBS LAYER # EXCEPTPGNET`. *Data_type*: float, optional
`-layer layer`	Specifies the layer for the OBS. *Data_type*: (layer), required
`-mask int_value`	Specifies the integer value of the mask (within the mask range) for the created OBS. *Default:* `0`, which means that the mask is uncolored. *Data_type*: uint, optional
`-polygon {x1 y1 x2 y2 x3 y3 ...}`	Specifies the polygon shape. *Data_type*: (point)+, optional
`-rects {{llx1 lly1 urx1 ury1} {llx2 lly2 urx2 ury2}...}`	Specifies the Obs rect list for adding obstruction geometries. *Data_type*: (rect)+, optional
`-spacing value`	Specifies the spacing value. *Data_type*: float, optional

Example

Following is an example of the `create_cell_obs` command usage:

```
create_cell_obs -layer M3 -cell tdsp_core -spacing 0 -rect {40 40 50 50}
```

Related Topics

`delete_cell_obs`

create_via

```
create_via
[-help]
-location {x y}
[-net net]
[-shape {blockring |
blockagewire | blockwire | corewire | drcfill | fillopc | fillwire | followpin | iowire | none | padring | ring | stripe}]
[-shield_net net]
[-status {routed | fixed | cover | shield}]
[-user_class value]
-via_def via_def
```

Adds a new special-route via instance to the design. A DB pointer is returned to the via, if successful. You can use the command at any point in the design flow.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `create_via` parameter. For a detailed description of the command and all its parameters, use the man command: `man create_via`
`-location {x y}`	Specify the placement location (in microns) of the via instance. *Data_type*: point, required
`-net net`	Specifies the name or pointer of the net to associate with the via instance. If not specified, the via is assigned as a floating fillwire via. *Data_type*: (net), optional
`-shape {blockring | blockagewire | blockwire | corewire | drcfill | fillopc | fillwire | followpin | iowire | none | padring | ring | stripe}`	
	Specifies the special route shape (`DEF SPECIALNETS wiring+ SHAPE`). If not given, the default is none unless `-net` is not specified. If `-net` is specified, the shape is set to `fillwire`. *Data_type*: enum, optional
`-shield_net net`	Specifies the name or pointer of the net to be shielded by the via instance. Only valid when `-status` is set to shield. *Data_type*: (net), optional
`-status {routed | fixed | cover | shield}`	
	Specifies the special route (`DEF SPECIALNETS`) wiring status. *Default*: `routed` *Data_type*: enum, optional
`-user_class value`	Specifies the user-defined string to be added to the via instance that is created. *Data_type*: string, optional
`-via_def via_def`	
	Specify the via-master name or pointer of the via-master to place. *Data_type*: (via_def), optional

Example

- The following command adds a via with via-master name of `my_via1`:
  ```
  create_via -via_def my_via1 -location {100.1 200.5} -net VSS -shape followpin
  ```
- To add a parameterized via you can look up the via-master with `create_via_definition` (in this case a 1x2 cut via with all the other parameters the default value).
  It will either return an existing via-master with the same parameters, or create it. Then use it to create an instance of the via.
  ```
  set myvia [create_via_definition -via_def_rule via1_rule -row_col {1 2}]
  create_via -via $myvia -pt {10.5 10.5} -net VS
  ```

create_via_definition

```
create_via_definition
[-help]
[-bottom_layer_masks int+]
[-cut_masks int+]
[-name via_name]
{{-via_rule via_def_rule
    [[-cut_size {x y}][-cut_spacing {x y}]
     [-top_enclosure {x y}]
     [-bottom_enclosure {x y}]
     [-top_offset {x y}][-bottom_offset {x y}]
     [-row_column {x y}]
     [-origin {x y}]
     [-pattern pattern]]} |
  {{-cut_layer layer -cut_rects {{x1 y1}{x2 y2}...}}
  {[-top_layer layer] {-top_rects {{x1 y1}{x2 y2}...} |
    -top_polygon {{x1 y1}{x2 y2}...}}}
  {[-bottom_layer layer] {-bottom_rects {{x1 y1}{x2 y2}...} |
    -bottom_polygon {{x1 y1} {x2 y2}...}}}}}
[-top_layer_masks int+]
```

Creates a via definition (master) from the specified parameters or list of shapes. The via masters created are design (DEF) vias and are for use in DEF SPECIALNETS wiring, unless the modify_ndr command is used to add them the list of valid vias for a NONDEFAULTRULE.

There are two types of via-master definitions:

- Parameterized vias that require a `-via_rule name`, and a set of row, column, spacing and enclosure parameters. These are equivalent to LEF/DEF VIAs defined with VIARULE GENERATE or OpenAccess standardViaDef. Applications normally check the via parameters for comparing and finding via-masters. But for LEF/DEF and reporting purposes, the via-master name is automatically generated using the `-via_rule name` as a prefix with a "_1", "_2", etc. suffix added to make it unique so it will not collide with any existing via-master names. The auto-generated name is only valid during the current session, and may be different after save/restore and opening it in another session. If a via-master already exists with the same parameters, the existing via-master is returned instead of creating a redundant via-master. Hence, this command can also be used to "lookup" any equivalent via-master previously created.
- Fixed vias that require a `-name` and a list of rectangles and polygons. These are equivalent to LEF/DEF VIAs using the RECT and POLYGON statements, or OpenAccess customViaDefs. The via name is the only way to lookup the via-master, and must be unique. The name cannot collide with an existing via-master. These types of via-masters are normally not recommended for design specific or DEF vias, because other blocks may create via-master names that collide. These types of vias are normally only in the library (LEF or OA techfile) so they are shared with all block of the same chip.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `create_via_definition` parameter. For a detailed description of the command and all its parameters, use the man command: `man create_via_definition`
`-bottom_enclosure {x y}`	
	Specifies the bottom layer enclosure around the cut array. The enclosure measures the distance from the cut-array-edge to the metal-edge that encloses the cut-array. Values are specified in microns. Requires `-via_rule` option for a parameterized via. *Default:* Derived from `-via_rule`. *Data_type*: point, optional
`-bottom_layer layer`	
	If not specified the first layer below the `-cut_layer` is assumed. Requires `-name` option for a fixed via. *Data_type*: (layer), optional
`-bottom_layer_masks int+`	
	Specifies the masks for the `rects` on the bottom layer for a `via_def`. The mask values and `rects` must be in the same order and there must be the same number of mask values as `rects`. *Data_type*: string, optional

`-bottom_offset {x y}`	
	Specifies the *x,y* offset for bottom layer in the via. After the “non-shifted via” is computed, the bottom layer rectangle is offset by adding the x,y to the `-origin` offset. Values are specified in microns. Requires `-via_rule` option for a parameterized via. *Default*: {0 0}. *Data_type*: point, optional
`-bottom_polygon {{x1 y1} {x2 y2}...}`	
	Specifies the polygon to create on the bottom layer. Values are specified in microns. Requires `-name` option for a fixed via. *Data_type*: (point)+, optional
`-bottom_rects {{x1 y1}{x2 y2}...}`	
	Specifies the list of rectangles to create on the bottom layer. Values are specified in microns. Requires `-name` option for a fixed via. *Data_type*: (point)+, optional
`-cut_layer layer`	
	Specifies the name or pointer of the cut layer. The top/bottom layers of the via are derived from the routing layers above/below the cut layer. Requires `-name` option for a fixed via. *Data_type*: (layer), optional
`-cut_masks int+`	
	Specifies the masks for the `rects` on the cut layer for a `via_def`. The mask values and `rects` must be in the same order and there must be the same number of mask values as `rects`. A mask pattern cannot be specified for parameterized via cuts. For parameterized via cuts, only three types of mask color patterns are allowed using the LEF VIA PATTERN statement - checker-board (default), alternating columns, and alternating rows. *Data_type*: string, optional
`-cut_rects {{x1 y1}{x2 y2}...}`	
	Specifies the list of rectangles to create on the cut layer. Values are specified in microns. Requires `-name` option for a fixed via. *Data_type*: (point)+, optional
`-cut_size {x y}`	Specifies the size (x=width, y=height) of the cut rectangles. Values are specified in microns and must be greater than 0. Requires `-via_rule` option for a parameterized via. *Default*: Derived from `-via_rule`. *Data_type*: point, optional
`-cut_spacing {x y}`	Specifies the spacing between the cuts. The spacing is measured from cut-edge to the next cut-edge. Values are specified in microns and must be greater than 0. Requires `-via_rule` option. *Default*: Derived from `-via_rule`. *Data_type*: point, optional
`-name via_name`	Specifies the name of via to create. For parameterized via definition creation, a new via name is derived from the `-via_rule` name unless `-name` is specified. The name cannot collide with any existing via-master names. *Data_type*: string, optional
`-origin {x y}`	Specifies the x,y offset for all shapes in the via. By default the 0,0 origin of the via is the center of the cut-array and the center of the enclosing metal rectangles. After the “non-shifted via” is computed, all the cut and metal rectangles are offset by adding these values. size (x=width, y=height) of the cut rectangles. Values are specified in microns. Requires `-via_rule` option for a parameterized via. *Default*: {0 0}. *Data_type*: point, optional
`-pattern pattern`	Specifies the cut pattern encoded as an ASCII string. This parameter is only required when some of the cuts are missing from the array of cuts. Requires `-via_rule` option for a parameterized via. *Data_type*: string, optional

`-row_column {x y}`	Specifies the number of cut rows and columns to create. Values must be greater than 0. Requires `-via_rule` option for a parameterized via. *Default*: {1 1}. *Data_type*: integer, optional
`-top_enclosure {x y}`	Specifies the top layer enclosure around the cut array. The enclosure measures the distance from the cut-array-edge to the metal-edge that encloses the cut-array. Values are specified in microns. Requires `-via_rule` option for a parameterized via. *Default*: Derived from `-via_rule`. *Data_type*: point, optional
`-top_layer layer`	If not specified the first layer above the `-cut_layer` is assumed. Requires `-name` option for a fixed via. *Data_type*: (layer), optional
`-top_layer_masks int+`	Specifies the masks for the `rects` on the top layer for a `via_def`. The mask values and `rects` must be in the same order and there must be the same number of mask values as `rects`. *Data_type*: string, optional
`-top_offset {x y}`	The x,y offset for top layer in the via. After the "non-shifted via" is computed, the top layer rectangle is offset by adding the x,y to the `-origin` offset. Values are specified in microns. Requires `-via_rule` option for a parameterized via. *Default*: {0 0}. *Data_type*: point, optional
`-top_polygon {{x1 y1}{x2 y2}...}`	
	Specifies the polygon to create on the top layer. Values are specified in microns. Requires `-name` option for a fixed via. *Data_type*: (point)+, optional
`-top_rects {{x1 y1}{x2 y2}...}`	
	Specifies the list of rectangles to create on the top layer. Values are specified in microns. Requires `-name` option for a fixed via. *Data_type*: (point)+, optional
`-via_rule via_def_rule`	
	Specifies the name or pointer of the via-rule (from LEF VIARULE GENERATE or OpenAccess standardViaDef) to use for the via master being created. Layers and default values are derived from the specified `via_rule`. When using `-via_rule`, a parameterized via master is created. *Data_type*: (via_def_rule), optional

Examples

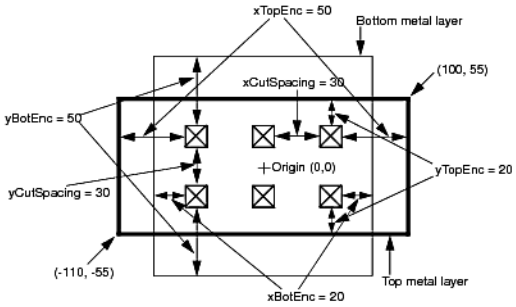
Example 1

Create a 1x2 cut via with default parameters.

```
set myvia [create_via_definition -via_rule via1_rule -row_column {1 2}]
dbGet $myvia.name #returns the via-master name
```

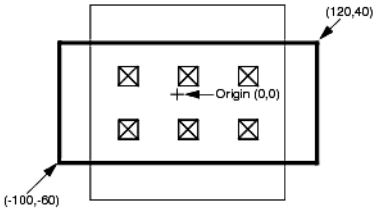
The following parameterized via-master describes a non-shifted via with two rows and three columns of via cuts.
In this example, all the size, spacing, and enclosure rules are given explicitly. In typical usage, they would not be specified and would use the default values in the via-rule.

```
create_via_definition -via_rule via1_rule -row_column {2 3} -cut_size {20 20} \
-cut_spacing {30 30} -bottom_enclosure {20 50} -top_enclosure {50 20}
```



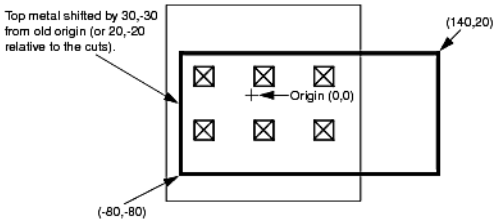
The same via rule with the –origin shifts all of the metal and cut rectangles by 10 in the x direction, and by -10 in the y direction.

```
create_via_definition -via_rule via1_rule -row_column {2 3} -cut_size {20 20} \
-cut_spacing {30 30} -bottom_enclosure {20 50} -top_enclosure {50 20} \
-origin (10 -10)
```



If the same via contains the following –origin and –top_offset parameters, all of the rectangles shift by 10, -10.
In addition, the top layer metal rectangle shifts by 20, -20, which means that the top metal shifts by a total of 30, -30.

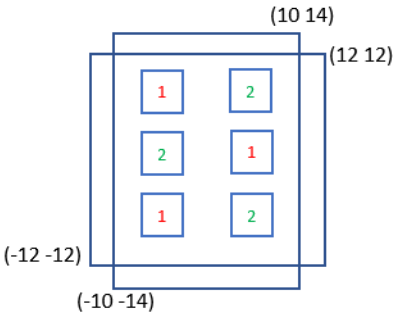
```
create_via_definition -via_rule via1_rule -row_column {2 3} -cut_size {20 20} \
-cut_spacing {30 30} -bottom_enclosure {20 50} -top_enclosure {50 20} \
-origin (10 -10) -top_offset {20 -20}
```



Example 2

The following command adds a `3x2` fixed via to the cut layer mask pattern `{1 2 1 2 1 2}`:

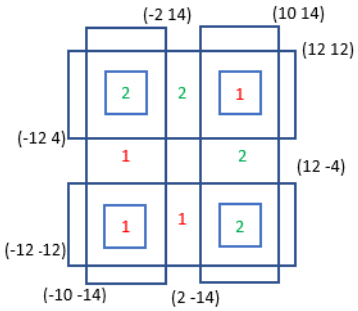
```
create_via_definition  -name my_fixed_via -cut_layer VIA23 \
-bottom_rects {{-10 -14 10 14}} \
-cut_rects {{-6 -10 -2 -6} {-6 -2 -2 2} {-6 6 -2 10} {2 -10 6 -6} {2 -2 6 2} {2 6 6 10}} \
-top_rects {{-12 -12 12 12}} \
-cut_masks {1 2 1 2 1 2}
```



Example 3

The following command adds a `2x2` fixed via to the bottom and top layer mask pattern `{1 2}` and cut layer mask pattern `{1 2 1 2}`:

```
create_via_definition -name my_fixed_via2 -cut_layer VIA23 \
-bottom_rects {{-10 -14 -2 14} {2 -14 10 14}} \
-cut_rects{{-8 -10 -4 -6} {-8 6 -4 10} {4 -10 8 -6} {4 6 8 10}} \
-top_rects {{-12 -12 12 -4} {-12 4 12 12}} \
-bottom_layer_masks {1 2} \
-top_layer_masks {1 2} \
-cut_masks {1 2 1 2}
```



create_virtual_shape_on_pg_wires

```
create_virtual_shape_on_pg_wires
[-help]
-layers {{virtual_layer, power_layer [mask_num]} ...}
[-name string]
[-pg_width_list {list_of_supported_width_values} | -pg_width_range {min_width max_width}]
```

Generates additional virtual shapes and creates markers on PG wires, when the existing trim shapes overlap with PG wires. The generated virtual shapes are routing blockages on the specified layer. These virtual shapes can be based on the output of the `write_def` and `write_stream` commands, and edited by the wire editing commands.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `create_virtual_shape_on_pg_wires` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man create_virtual_shape_on_pg_wires`
`-layers {{virtual_layer, power_layer [mask_num]} ...}`	Specifies the pair of layers, which need to create the virtual shapes. You can specify the optional argument `mask_num` with this parameter to select the PG wires. *Data_type*: string, required
`-name string`	Specifies the name of the marker shape to be generated. *Default*: `defLayerBlkName` *Data_type*: string, optional
`-pg_width_list {list_of_supported_width_values}`	Creates virtual shapes on PG pin only for the specified width values. **Note:** This parameter cannot work with the `-pg_width_range` parameter. It means that you can define either a range of width, or a list of width. You can use the `-pg_width_list` parameter with the `-layer` and `-name` parameters. *Data_type*: string, optional
`-pg_width_range {min_width max_width}`	Specifies the PG width range in which the virtual shapes are needed. *Default*: All the PG wires. *Data_type*: string, optional

decompress_special

```
decompress_special
[-help]
[-area {x1 y1 x2 y2}]
[-layers layer+]
[-nets net+]
```

Decompresses Power-Grid (PG) objects within the specified area, layer, or net. This command decompresses only the overlapped part in case of partial overlapping. The rest of the part is regrouped.

Parameters

`-help`	Prints a brief description that includes the type and default information for each `decompress_special` parameter. For a detailed description of the command and all its parameters, use the man command: `man decompress_special`
`-area {x1 y1 x2 y2}`	Specifies the coordinates for the area from which PG needs to be decompressed. The `decompress_special` command decompresses only those special wires and special vias, which exist within the area specified by this parameter. *Default*: All PG structures are decompressed. *Data_type*: rect, optional
`-layers layer+`	Specifies the layers on which PG decompression is needed. This parameter requires either LEF layer names or a routing layer index value (for example, `1` for the first routing layer, `2` for the second routing layer, and so on). *Data_type*: (layer)+, optional
`-nets net+`	Specifies the nets from which PG decompression is needed. *Data_type*: (net)+, optional

Example

The following command decompresses `%s special_wires` and `%s special_vias`:

```
decompress_special -area {100 100 120 120} -layer Via56
```

Related Topics

`compress_special`

delete_cell_obs

```
delete_cell_obs
[-help]
-cells {base_cell | design | module}+
{-obs_shapes shape+ | [{-rects {{llx1 lly1 urx1 ury1} {llx2 lly2 urx2 ury2}...} | -polygon {x1 y1 x2 y2 x3 y3 ...}] -layers layer+}
```

Helps in deleting an obstruction geometry or library obstruction for a cell on the specified layer. This command deletes only the obstruction geometry added by the `add_cell_obs` command or library obstruction (`SPACING 0 OBS`).

Parameters

`-help`	Prints a brief description that includes the type and default information for each `delete_cell_obs` parameter. For a detailed description of the command and all its parameters, use the man command: `man delete_cell_obs`
`-cells {base_cell | design | module}+`	Specifies the cell name or DPO name. *Data_type*: (base_cell|design|module)+, required
`-layers layer+`	Specifies the layer name or DPO name. *Data_type*: (layer)+, optional
`-obs_shapes shape+`	Specifies the cell obstruction shape pointers. **Note:** This parameter must be used with the `-cells` parameter, and cannot be specified with any other parameter except than `-help`. *Data_type*: (shape)+, optional
`-polygon {x1 y1 x2 y2 x3 y3 ...}`	Specifies the polygon shape. *Data_type*: (point)+, optional
`-rects {{llx1 lly1 urx1 ury1} {llx2 lly2 urx2 ury2}...}`	Specifies the Obs rect list for deleting obstruction geometries. *Data_type*: (rect)+, optional

Examples

Following is an example of using the `-obs_shapes` parameter for specifying the cell obstruction shape pointer:

- The `BUFX1` cell has multiple obs shapes on different layers:
  ```
  get_db base_cell:BUFX1 .obs_layer_shapes.shapes
  shape:0x7f7f50db3b00 shape:0x7f7f50db3ae0 shape:0x7f7f50db3ac0 shape:0x7f7f50db3aa0 shape:0x7f7f50db3a80
  ```
- The following command deletes the first two shapes in the list:
  ```
  delete_cell_obs -obs_shapes {shape:0x7f7f50db3b00 shape:0x7f7f50db3ae0} -cells BUFX1
  ```

delete_floating_nets

```
delete_floating_nets
```

The `delete_floating_nets` command deletes all the nets with zero terms except for assign nets and p/g nets. It supports zero-term net deletion in multiple instantiated cells.
This command takes no parameters and can be used whenever the design is read in.

After the command is executed, it reports the following:

- Total number of dangling nets `nrDnet`
- Total number of deleted dangling nets `nrDdnet`
- Total number of nets `ntNet`
- Ratio of `nrDdnet` over `nrNet`
- Ratio of `nrDnet` over `nrNet`
- `nrDdnet` over `nrDnet`

Related Topics

`report_floating_nets`

delete_global_net_connections

```
delete_global_net_connections [-help]
```

Resets the global net as well as all of the tie-high and tie-low nets. Use this command after you issue the `globalNetConnections` command.

Parameters

`-help`	Prints a brief description of the `delete_global_net_connections` command usage. For a detailed description of the command, use the `man` command: `man delete_global_net_connections`

Related Topics

Power Planning and Routing chapter in the *Innovus User Guide*.

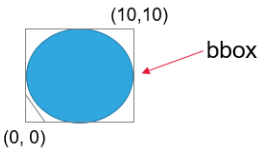
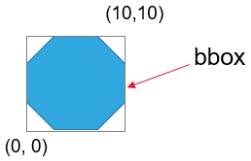
get_computed_shapes

```
get_computed_shapes
[-help]
[-dbu]
[-step step]
[-output {polygon|rect|hrect|area}]
shape_list [AND shape_list | ANDNOT shape_list | OR shape_list | XOR shape_list
| INSIDE shape_list | OUTSIDE shape_list | ENCLOSE shape_list | STRADDLE shape_list
| BBOX | HOLES | NOHOLES | BBOX_CIRCLE | BBOX_OCTAGON | MOVE {dx  | SIZE value | SIZEX value | SIZEY value] ...
```

Performs geometric (SIZE and BBOX) and logical (such as AND and ANDNOT) operations on shape (polygon or a rectangle or a list of rectangles and polygons).

Parameters

`-help`	Prints a brief description that includes the type and default information for each `get_computed_shapes` parameter. For a detailed description of the command and all its parameters, use the man command: `man get_computed_shapes`
`-dbu`	Specifies values in database units, specified in integers. All `shape_list` values as well as values for MOVE, SIZE, and step should The output units are the same as the input units. *Default:* Specifies values in user units, in microns. *Data_type*: bool, optional
`-step`	Specifies step size if output format is rect, required to convert non-orthogonal shapes into series of rectangles. *Default*: minwidth value of first routing layer. The option is coord, and is optional. *Data_type*: coord, optional
`-output` *polygon|rect|hrect|area*	
	Specifies the output format. • `polygon`: Polygon shape • `rect`: Vertical rectangles **Note:** This value works only for the orthogonal shapes. • `hrect`: Horizontal rectangles **Note:** This value works only for the orthogonal shapes. • `area`: Total area of final shape *Default:* `rect` (string) *Data_type*: enum, optional
`shape_list`	A polygon or rectangle or a list of polygon and rectangle. • polygon: `{{x y} {x y} {x y} ... }`; • rect: `{x1 y1 x2 y2}` This option is a required list. *Data_type*: list, required
`AND` *shape_list*	Is a binary operator that is the intersection of `shape_lists`.
`ANDNOT` *shape_list*	Is a binary operator which defines the initial `shape_list` minus `shape_list`.
`OR` *shapelist*	Is a binary operator of the union of shapelists.
`XOR` *shape_list*	Is a binary operator of the `shape_list`: OR minus AND. *Data_type*: string, optional

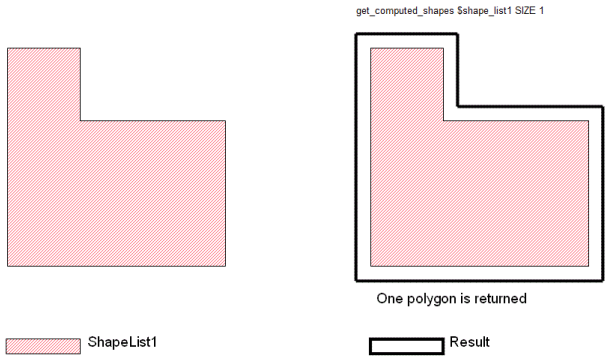
`INSIDE shape_list`	Is a binary operator for initial shapes that are inside (including touching) of one of the shapes from the second list.
`OUTSIDE shape_list`	Is a binary operator for initial shapes that are outside (including touching) of one of the shapes from the second list.
`ENCLOSE shape_list`	Is a binary operator for initial shapes that completely enclose at least one shape from the second list.
`STRADDLE shape_list`	Is a binary operator for initial shapes that are partially inside and partially outside at least one shape from the second list.
`MOVE {dx dy}`	Is a unary operator that moves the shape {`dx dy`}.
`BBOX`	Is a unary operator that computes the bounding box of `shape_list`. *Data_type*: string, optional
`BBOX_CIRCLE`	Is a unary operator that creates a circle shape. For example, the following command creates a circle within the specified coordinat `get_computed_shapes {0 0 10 10} BBOX_CIRCLE -output {polygon}` Following is the output of the above command: 
`BBOX_OCTAGON`	Is a unary operator that creates an octagon shape. For example, the following command creates an octagon within the specified co `get_computed_shapes {0 0 10 10} BBOX_OCTAGON -output {polygon}` Following is the output of the above command: 
`HOLES`	Is a unary operator that gives list of new shapes that fill holes in the original `shape_list`.
`NOHOLES`	Is a unary operator that gives list of new shapes that include filled holes in the original `shape_list`. This is same as `get_computed_shapes shape_list HOLES` OR `shape_list`.
`SIZE value`	Is a unary operator that increases the size of `shape_list` by `value`.
`SIZEX value`	Is a unary operator that increases the size of the shapes in the list in the X-direction by `value`. This parameter works only for the ort `-step` parameter to convert the shape into series of rectangles first.
`SIZEY value`	Is a unary operator that increases the size of the shapes in the list in the Y-direction by `value`. This parameter works only for the ort `-step` parameter to convert the shape into series of rectangles first.

Note: All shapes processed by `get_computed_shapes` must be non-zero area. If a *shape_list* contains a list of rectangles and one of them is zero-area, it will be dropped.
Also, during negative sizing or other operations, if a zero-area shape is created, it will be dropped from the output (or intermediate results).

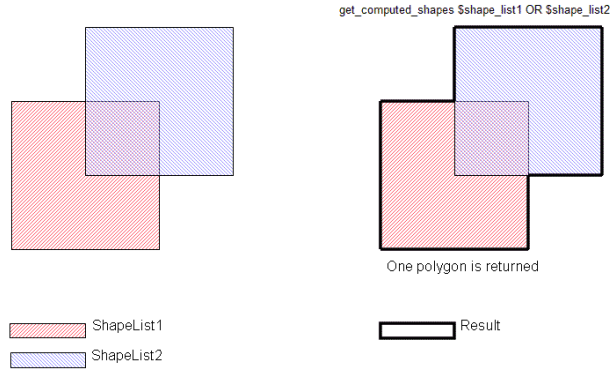
For example, a rectangle that is 4x4 or smaller will disappear when using `get_computed_shapes $rect SIZE -2 SIZE 2` while shapes larger than 4 will be returned to their original state.
The same is true for protrusions with width less than 4.

Examples

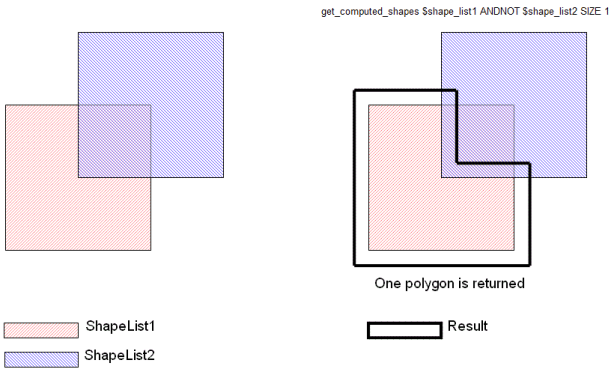
- **Unary**:



- **Binary**:



- **Combination**:



Example: Area calculation

`get_computed_shapes $shapes1 AND $shapes2`
Returns intersection (as a list of rectangles) between the shapes in the `$shapes1` and `$shapes2` lists.

`get_computed_shapes -output area $shapes1 AND $shapes2`
Returns intersection area between the shapes in the `$shapes1` and `$shapes2` lists.

get_edge

```
get_edge
[-help]
[-clockwise | -counterclockwise]
[{shape_list} [-direction direction] [-db_units]]
[-index index_number]
```

Returns edges of a polygon of a shape along with the direction list (N, S, E, W and NE, NW, SE, SW). This helps you identify not only all the edges but also the direction in which the edges are pointing. At a time you can observe one polygon with 'n' number of points.

Parameters

`-help`	Prints a brief description that includes the type and default information for each `dbedge` parameter. For a detailed description of the command and all its parameters, use the man command: `man get_edge`
`-clockwise`	Sorts the returned list of edges clockwise. *Data_type*: bool, optional
`-counterclockwise`	Sorts the returned list of edges counterclockwise. *Data_type*: bool, optional
`-db_units`	Returns the specified values in database units. Default is `microns`. *Data_type*: bool, optional
`-direction {n s e w ne nw se sw}`	Specifies the types of edges needed. This can be n, s, e, w, ne, nw, se, sw. It can also be a list. For example: `get_edge -direction {n s nw ne} {10 20 30 40}` *Data_type*: (enum)+, optional
`-index index_number`	Specifies the index number of the edge to be returned. The index number starts with `0` as the first edge specified in the shape list, and increases gradually along with the shape list. This number may range from `0` to '*number_of_edges* `- 1`'. *Data_type*: int, optional

`{shape_list}`	Can be a rect like `{x1 y1 x2 y2}` or a polygon like `{{x1 y1}{x2 y2}{x3 y3}.....}`. Only one rect or polygon is accepted.
	Examples
	Using a simple L-shape polygon
	`set shape_list {{0 0} {2 0} {2 1} {1 1} {1 2} {0 2}}`
	• All edges
	`get_edge $shape_list`
	`{{S{0 0}{2 0}} {E{2 0}{2 1}} {N{2 1}{1 1}} {E{1 1}{1 2}} {N{1 2}{0 2}} {W{0 2}{0 0}}}`
	• Only E facing edges
	`get_edge -direction E $shape_list`
	`{{E{2 0} {2 1}} {E{1 1} {1 2}}}`
	The format of the output is
	`{{<direction> <edge_vertex_1> <edge_vertex_2 }....}`
	Data_type: string, required

get_obj_in_area

```
get_obj_in_area
[-help]
[-areas {{llx1 lly1 urx1 ury1} {llx2 lly2 urx2 ury2} ...}
[-bbox_overlap]
[-dbu]
[-layers {layer1 layer2 ...} [-strict_via_inst_layer_check]]
[-obj_type {bump bus_guide inst inst_all_shapes pin port port_shape marker net place_blockage patch_wire pg_pin route_blockage routes
resize_blockage row special special_via special_wire via wire what_if_wire what_if_via partition}]
[-abut_only] | [-enclosed_only] | [-overlap_only]
[-polygons {x11 y11 x12 y12 x13 y13 ...}]
```

Returns a list of objects based on an area query without needing to select them. This command supports all types of routing blockages such as partial, fill, and slot.

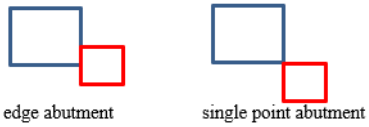
Parameters

`-help`	Prints a brief description that includes type and default information for each `get_obj_in_area` parameter. For a detailed description of the command and all its parameters, use the man command: `man get_obj_in_area`
`-abut_only`	Returns only those objects that abut the search area. No part of the object polygon overlaps the search area, but at least one point or line of intersection occurs between the object polygon and the search area. A single point of abutment (such as a corner or crossing edges) is also considered as abutted. When multiple areas are specified, the object is returned as long as it abuts one of the search areas. *Data_type*: bool, optional
`-areas {{llx1 lly1 urx1 ury1} {llx2 lly2 urx2 ury2} ...}`	
	Specifies the search areas, where `ll` = lower left, `ur` = upper right. By default, all the objects overlapping or abutting these areas are returned, unless `-abut_only`, `-enclosed_only`, or `-overlap_only` is specified. The polygon shape of the object is used for the overlap check. The units are in microns (unless `-d` is specified). The input can be in either a single `rect {llx1 lly1 urx1 ury1}`, or a list of `rects {{llx1 lly1 urx1 ury1} {llx2 lly2 urx2 ury2}}`. A `rect` using "two points" format like `{{{llx1 lly1} {urx1 ury1}} {{llx2 lly2} {urx2 ury2}}}` is also allowed. But input format like `{llx1 lly1 urx1 ury1 llx2 lly2 urx2 ury2}` or `{{llx1 lly1} {urx1 ury1} {llx2 lly2} {urx2 ury2}}` is illegal and not allowed. The resulting area to check for overlap can be a rectangle, line, or a point. When one object intersects multiple search areas, it will only be returned once. *Data_type*: (rect)+, optional
`-bbox_overlap`	
	Uses bounding-box of the object rather than the polygon shape of the object for overlap for overlap, abut, or enclosure check. For example, an `inst` that is of L-shape, will not be returned if the `-areas` parameter overlaps only the upper-right of the bounding-box without touching any part of the L-shaped, unless the `-bbox_overlap` parameter is included. *Data_type*: bool, optional
`-dbu`	Specifies the `-areas` values that are in database units rather than microns. *Default*: `um` *Data_type*: bool, optional
`-enclosed_only`	Returns only those objects, which are completely enclosed inside the search area. The object polygon is fully enclosed by the search area. No part of the object polygon sticks outside the search area, although it may abut the search area edges. When multiple areas are specified, the object will be returned as long as it is enclosed in one of the search areas. *Data_type*: bool, optional
`-layers {layer1 layer2 …}`	

	Returns all the objects with the layer information for the specified layers. This parameter applies only to object types: `port_shape`, `patch_wire`, `route_blockage`, `regular`, `special`, `special_via`, `special_wire`, `via`, `wire`, `what_if_wire`, and `what_if_via`. A `via`, `special_via`, or `what_if_via` object will be returned if any of the via's bottom layer, cut layer, or top layer matches with one of the specified layers. Other object types ignore this parameter and will always be returned. *Data_type*: (layer)+, optional
`-obj_type` *objTypeList*	
	Specifies a list of object types to search for overlaps in the area. Multiple object types can be searched at the same time. The objects currently supported include: `bump bus_guide inst inst_all_shapes pin port port_shape marker net place_blockage patch_wire pg_pin route_blockage routes resize_blockage row special special_via special_wire via wire what_if_wire what_if_via partition` Here: • `routes` is the union of `wire`, `patch_wire`, and `via`. • `special` is the union of `special_via` and `special_wire`. Objects with multiple shapes or non-rectangular shapes use a polygon to check for overlap, as described below. • `bump` has no placement overlap area, so it uses the polygon of all the OBS and PIN shapes. • `inst` uses the polygon of its placement overlap checks (for example the LEF SIZE or OVERLAP shapes), including any placement halo (if there is any) but not any routing halo. OBS shapes or PIN shapes (like standard-cell power-pins) that stick outside the placement overlap area are not included. • `inst_all_shapes` is the same as `inst`, except that it includes all OBS shapes and PIN shapes even if they stick outside the placement overlap area. • `pin` uses the polygon of all the pin-shapes of the `pin`. • `pg_pin` uses the polygon of all the pin-shapes of the `pg_pin`. • `port_shape` uses the polygon of all the shapes of the top-level `port_shape`. • `net` uses the polygon of all the regular routing shapes on the net. Pin-shapes or special-route shapes attached to the net are not included. • `port` uses the polygon of all the pin-shapes of the top-level port. • `port_shape` uses the polygon of all the pin-shapes of the top-level port shape. • `via` uses the polygon of all the shapes of the via. • `special_via` uses the polygon of all the shapes of the via. • `partition` uses the polygon of all the shapes of the partition. *Default*: `inst` *Data_type*: (enum)+, optional
`-overlap_only`	Returns only those objects, which are partially overlapped with the search area. Some part of the object polygon overlaps the search area, whereas some of its part is outside the search area. A single point of contact is not considered as an overlap. When multiple areas are specified, the object is returned as long as it overlaps with one of the search areas. *Data_type*: bool, optional
`-polygons {`*x11 y11 x12 y12 x13 y13 ...*`}`	
	Specifies a polygon search area. By default, all the objects overlapping or abutting this area are returned on using the `-polygons` parameter, unless the `-abut_only`, `-enclosed_only`, or `-overlap_only` parameter is specified. The units are in microns (unless `-d` is specified, which indicates the database units). The format can be in either like `{x11 y11 x12 y12 x13 y13 ...}` or `{{x11 y11} {x12 y12} {x13 y13} ...}`. The polygon must be well-formed, not have zero area, and not have crossing edges. *Data_type*: polygon, optional
`-strict_via_inst_layer_check`	
	Returns only those objects on the specified layer, which match with one of the via's bottom, cut, or top layers. This parameter does not return the objects of the via on the unmatched layers. It works only for `special_via` and `via`. *Data_type*: bool, optional

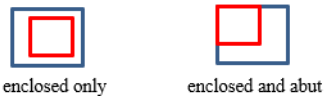
Examples

- The following command returns all the instances and routing-blockages on the `metal6` and `metal7` layers that are partially overlapped, abutted, or enclosed in the given triangle area:
  ```
  get_obj_in_area -obj_type {inst route_blockage} -polygons {100 100 110 110 100 110} -layers {Metal6 Metal7}
  ```
- The following command returns all the instances and routing-blockages on the `metal6` and `metal7` layers that are partially overlapped, abutted, or enclosed in the given triangle area using the bounding-box of the objects:
  ```
  get_obj_in_area -obj_type {inst route_blockage} -polygons {100 100 110 110 100 110} -layers {Metal6 Metal7}} -bbox_overlap
  ```
- The following command returns all the instances and regular wires on the `metal6` and `metal7`layers that are abutted in the give search areas:
  ```
  get_obj_in_area -obj_type {inst wire} -areas {{0 0 100 100} {150 100 200 200}} -layers {Metal6 Metal7}} -abut_only
  ```
- The following command returns all the instances and regular wires on the `metal6` and `metal7`layers that are abutted in the given search areas using the bounding-box of the objects:
  ```
  get_obj_in_area -obj_type {inst wire} -areas {{0 0 100 100} {150 100 200 200}} -layers {Metal6 Metal7}} -abut_only -bbox_overlap
  ```
- Following are the figures to illustrate each position relation. The instances presented by the red rectangles are returned with the specified parameter when the search area presented in the blue rectangle are specified.
  - **`-abut_only`**

  - **`-overlap_only`**

  - **`-enclosed_only`**

Note:

1. If none of the above three parameters is specified, all the instances presented by the red rectangles will be returned, by default.
2. For querying object with multi-shape (such as `port_shape`) in multiple search areas, if one of its shapes is enclosed/abutted/overlapped with any one of the specified areas, the object will be returned.

get_rect

```
get_rect
[-help]
rect {-ll | -ur | -llx | -lly | -urx | -ury | -size | -dx | -dy | -width | -length | -center | -area}
```

Returns the coordinates of the points of a rectangle. When you need to find the coordinates of a rectangle, in order to either move the rectangle or reshape the rectangle, the command helps you find the coordinates.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `get_rect` parameter. For a detailed description of the command and all its parameters, use the man command: `man get_rect`
rect {-ll | -ur | -llx | -lly | -urx | -ury | -size | -dx | -dy | -width | -length | -center | -area}	
	Specified rectangle to access values from. • `-area`: Returns the area of the rectangle. • `-center`: Returns the center of the rectangle. • `-length`: Returns the larger of the `-dx` and `-dy` values. • `-ll`: Returns the lower left point of the rectangle. • `-llx`: Returns the lower left x coordinate of the rectangle. • `-lly`: Returns the lower left y coordinate of the rectangle. • `-dx`: Returns the x difference between the larger and smaller x values of the rectangle. • `-dy`: Returns the y difference between the larger and smaller y values of the rectangle. • `-ur`: Returns the upper right point of the rectangle. • `-urx`: Returns the upper right x coordinate of the rectangle. • `-ury`: Returns the upper right y coordinate of the rectangle. • `-size`: Returns the size of the rectangle from lower left to upper right. • `-width`: Returns the smaller of the `-dx` and `-dy` values. For *rect*: *Data_type*: rect, required For all other parameters mentioned above: *Data_type*: bool, optional

Examples

- `-area {8.1 9.2 10.3 11.4} returns 4.84`
- `-center {8.1 9.2 10.3 11.4} returns 9.2 10.3`
- `-length: get_rect -length {8.1 9.2 10.3 11.4}` returns: `2.2`
- `-ll: get_rect -ll {8.1 9.2 10.3 11.4}` returns: `8.1 9.2`
- `-llx: get_rect -llx {8.1 9.2 10.3 11.4}` returns: `8.1`
- `-lly: get_rect -lly {8.1 9.2 10.3 11.4}` returns: `9.2`
- `-dx: get_rect -dx {8.1 9.2 10.3 11.9}` returns: `2.2`
- `-dy: get_rect -dy {8.1 9.2 10.3 11.9}` returns: `2.7`
- `-ur: get_rect -ur {8.1 9.2 10.3 11.4}` returns: `10.3 11.4`
- `-urx: get_rect -urx {8.1 9.2 10.3 11.4}` returns: `10.3`
- `-ury: get_rect -ury {8.1 9.2 10.3 11.4}` returns: `11.4`
- `-size: get_rect -size {8.1 9.2 10.3 11.4}` returns `2.2 2.2`
- `-width: get_rect -width {8.1 9.2 10.3 11.4}` returns: `2.2`

get_transform_shapes

```
get_transform_shapes
[-help]
[-db_units]{-inst instPtr | -hier_inst instPtr | {-cell cellPtr -orient orientEnum -pt {x y}}}
{-local_pt list | -global_pt list}
```

Takes coordinates local to a cell and returns them in context with the global design space. You can specify a specific instance of a cell in the design, or you can specify a library cell, its location in the design, and instruct the software to perform a "what-if" translation of coordinates within that cell.

Parameters

-help	Pints a brief description that includes the type and default information for each `get_transform_shapes` parameter. For a detailed description of the command and all its parameters, use the man command: `man get_transform_shapes`
`-cell` *cellPtr*	Specifies a pointer to a library cell. You must specify the `-pt` and `-orient` parameters with `-cell`. *Data_type*: (base_cell|module|design), optional
`-db_units`	Specifies values in database units, specified in integers. This is an optional parameter. If you use the `-db_units` option, the Boolean value is 1, so the value will be displayed in database units. If you don't specify the `-db_units` option, the Boolean value is 0, the default. *Default*: Specifies values in user units, in microns *Data_type*: bool, optional
`-global_pt` *globalPts*	Specifies the global coordinates to convert into local coordinates. *Data_type*: string, required
`-hier_inst`	Specifies the hierarchical instance of a database object. This option is mandatory. *Data_type*: (hinst), required
`-inst` *instPtr*	Specifies a pointer to a specific instance/bump of a cell in the design. The software derives the coordinates and the orientation for the cell instance from the design. *Data_type*: (inst | bump), required
`-local_pt` *list*	Specifies the local coordinates to convert into global coordinates. *Data_type*: string, required
`-orient` *orientEnum*	Specifies the orientation of the cell specified with `-cell`. You must specify the `-cell` and `-pt` parameters with `-orient`. Valid orientation values are: `r0`, `r90`, `r180`, `r270`, `mx`, `mx90`, `my`, and `my90`. *Data_type*: enum, optional
`-pt` *x y*	Specifies the coordinates for the cell specified with `-cell`. You must specify the `-cell` and `-orient` parameters with `-pt`. *Data_type*: point, optional

Examples

- Assume that a shape exists on *M1* within a NAND2 cell, from (`1 1`) to (`2 2`). If an instance of the NAND2 cell is placed at (`10 10`), with an `r0` (N) orientation, the following command converts the local points (`1 1 2 2`) of the NAND2 cell into their global coordinates:

  ```
  get_transform_shapes -cell [get_db base_cells -if {.base_name == NAND2}] -pt {10 10} -orient r0 -local_pt {1 1 2 2}
  ```

  The software returns the following information:

  ```
  {11 11 12 12}
  ```

- Assume that the NAND2 cell placed at (`10 10`), with an `r0` (N) orientation is named `i1`. The following command converts the local points (`1 1 2 2`) of `i1` into their equivalent global coordinates:

  ```
  get_transform_shapes -inst [get_db insts -if {.name == i1}] -local_pt {1 1 2 2}
  ```

The software returns the following information:

```
{11 11 12 12}
```

report_floating_nets

```
 report_floating_nets
[-help]
[-out_file file_name]
```

Reports the number of nets with zero terms, including the PG nets. This command can also be used to report the number of dangling nets, which could or could not be removed by the `delete_floating_nets` command. The command reports the following information, when used:

- Total number of nets
- Number of dangling nets
- Number of removable dangling nets
- Number of non-removable dangling nets

The removable and non-removable nets are reported based on the usage of the `delete_floating_nets` command. Those dangling nets, which could not be removed by the `delete_floating_nets` command, are reported in the output file due to multiple possible reasons, such as the dangling net:

- Is a physical net.
- Is a PG net.
- Is a bus bit net.
- Is an assign net.
- Has multiple module ports and at least one port is associated with some constraints.
- Has some constraints, such as in an ILM module or some timing constraint.

Note: If a dangling net cannot be deleted due to any reason, only the first reason listed above will be reported in the output file.

Parameters

`-help`	Prints a brief description that includes the type and default information for each `report_floating_nets` parameter. For a detailed description of the command and all its parameters, use the man command: `man report_floating_nets`
`-out_file file_name`	Reports the list of dangling nets that cannot be deleted by the `delete_floating_nets` command, along with its reasons. *Data_type*: string, required

Examples

- The following command reports the information about the nets listed in the `dangling_net.rpt` file:

```
report_floating_nets -out_file dangling_net.rpt
```

- Following is the output of the above command:

```
############################################################
# Generated by: Cadence Innovus 19.10-p002_1
# OS: Linux x86_64(Host ID sjfsb473)
# Generated on: Sat Aug 18 11:14:18 2018
# Design: design_name
# Command: report_floating_nets -out_file dangling_nets.rpt
############################################################

Total number of nets: 123456
The number of dangling nets: 26
Removable dangling nets: 22
Non-removable dangling nets: 4
Is physical net: 2
Is P/G net: 0
Is bus net: 1
Is assign net: 0
Has module port:0
Has constraints: 1


Reasons for the net not being removed:
net:xxxx is physical net ## net:xxxx here is the DPO name of the net
net:xxxx is physical net
net:xxxx is bus net
net:xxxx has constraints
```

Related Topics

`delete_floating_nets`

report_metal_stack

```
report_metal_stack
[-help]
[-non_preferred_direction]
```

Reports the metal stack configuration and routing track information. For each routing layer, this includes the layer name, preferred routing direction, routing offset (offset of routing track from 0), routing pitch (space between each routing track), min-width DRC rule, min-spacing DRC rule, and the routing default-width (wire width of the default routing rule). Note that for a horizontal direction track, the offset and pitch are in the Y direction (for example, the Y offset from 0, and the Y pitch of horizontal tracks). This command only supports uniform track-patterns with a single pitch value, and will not report non-uniform pitch routing track patterns that can be created with `add_tracks` or `create_tracks`.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `report_metal_stack` parameters. For a detailed description of the command and all its parameters, use: `man report_metal_stack`
`-non_preferred_direction`	By default, the `report_metal_stack` command displays the metal stack and routing information only related to the preferred routing direction. When the `-non_preferred_direction` parameter is used, the information related to the non-preferred routing direction is also displayed. *Data_type*: bool, optional

Examples

- The following command displays the metal stack and routing information:
  ```
  report_metal_stack
  ```

  The following is an example of the output of the above command:

  ```
  ================================================================================
  Layer   Direction   Offset    Pitch      Min_Spacing  Min_Width   Default_Width
  ================================================================================
  M1      horizontal  0.000000  0.040000   0.020000     0.020000     0.020000
  M2      vertical    0.000000  0.057000   0.020000     0.037000     0.037000
  M3      horizontal  0.000000  0.040000   0.020000     0.020000     0.020000
  M4      vertical    0.000000  0.048000   0.020000     0.028000     0.028000
  ```

- The following command adds in the non-preferred routing direction information:
  ```
  report_metal_stack -non_preferred_direction
  ```

  The following is the output of the above command:

  ```
  ============================================================================================
  Layer  Direction     Preferred Offset    Pitch      Min_Spacing  Min_Width  Default_Width
  ============================================================================================
  M1      horizontal   yes       0.000000  0.040000   0.020000     0.020000     0.020000
  M1      vertical     no        0.000000  0.040000   0.000000     0.000000     0.060000
  M2      vertical     yes       0.000000  0.057000   0.020000     0.037000     0.037000
  M2      horizontal   no        0.000000  0.057000   0.000000     0.000000     0.060000
  M3      horizontal   yes       0.000000  0.040000   0.020000     0.020000     0.020000
  M3      vertical     no        0.000000  0.040000   0.000000     0.000000     0.060000
  M4      vertical     yes       0.000000  0.048000   0.020000     0.028000     0.028000
  M4      horizontal   no        0.000000  0.048000   0.000000     0.000000     0.060000
  ```

Related Topics

- `add_tracks`
- `create_tracks`

report_net_wires

```
report_net_wires
[-help]
[-file fileName]
[-net netName]
```

Displays information about the regular net wires. If the `-net` parameter of this command is not specified, the wires of all the nets are displayed. If the `-file` parameter of this command is not specified,, the wire information on the screen (console) and log file are displayed. The information about the net regular wires is categorized as:

- Node Information
- Edge Information

Node Information

It includes XY location (like `Loc=(2.3, 52.11)`), metal layer (like `Layer=M1(2)`), number of edges on the current node (like `Edges=2`), tie special node name (like `Tie Special`), and global route node name (like `Global Route`). If the node has a terminal attached, the node information also includes the term information, like `Term = a/b/c`, `IO Pin = name`, or `PG Term = x/y/z`.

Edge Information

It includes node number (like `Node3 -> Node5`), edge type (like `Type=HV Wire`), and edge status (like `Status=Routed`). The edge status list includes `None`, `Routed`, `Fixed`, `Cover`, `Shield`, `No Shield`, `Global`, `Assigned`. For different edge types, the edge information reports different edge attributes. Following is the list of various edge types and their attributes:

- `HV Wire:`
  `NDR=name` (non-default rule name)
  `Width=3.3` (wire width)
  `Begin Extension=3.3` (wire beginning extension)
  `End Extension=3.3` (wire ending extension)
  `Mask=1` (color/mask value)
- `Via:`
  `Via Cell=name` (via cell name)
  `NDR=name` (non-default rule name)
  `Top Mask=1` (top metal layer's color/mask value)
  `Cut Mask=1` (cut layer's color/mask value)
  `Bottom Mask=1` (bottom metal layer's color/mask value)
- `Patch:`
  `Box=(100.0,200.0,223.6,250.4)` (patch box coordinates)
  `Mask=1` (patch color/mask value)
  `Attr=304` (patch attributes value)
  If it is a non-routing patch, it will also report trim metal information as shown below:
  `Trim Metal Box=(10.0, 10.0, 11.9, 11.9)` (trim metal box coordinates)
  `Mask=1` (trim metal color/mask value)
  `Layer=name` (trim metal layer name)
- `45 Degree:`
  `NDR=name` (non-default rule name)
  `Width=3.3` (wire width)
  `Mask=1` (color/mask value)
- `Zero Length`
- `Term Connect`
- `Virtual:`
  `Zero Length`, `Term Connect`, and `Virtual` have no extra attribute information. Following is an example of a single net:
  Wires for the `net1` net:
  `Node0: Loc=(44.102, 55.203), Layer=PO(2), Edges=1, Global Route`

```
Term = inst1/b
Node1: Loc=(44.102, 66.304), Layer=Po(2), Edges=1
IO Pin = net1
Edge0: (Node0->Node1), Type=HV Wire, Status=Routed, NDR=ndr1, Width=0.056, Begin Extension=0.056, End Extension=0.056, Mask=1
```

Parameters

`-help`	Prints a brief description that includes the type and default information for each `report_net_wires` parameter. For a detailed description of the command and all its parameters, use the man command: `man report_net_wires`
`-file` *`fileName`*	Specifies the name of the file.
`-net` *`netName`*	Specifies the name of the net.

report_preserves

```
report_preserves
[-help]
[-computed]
[-dont_touch | -dont_use]
[-obj_type {design | hinst | module | inst | net | hnet | pin | hpin | base_cell}]
```

By default, this command displays all the preserves that affect optimization set in the library. This includes all the `.dont_touch*`, `.read_only*`, `.is_ilm`, `.dont_use*`, and `.use_cells` attributes that have a non-default value. The command does not show the computed attributes that include inheritance from their parents such as `.dont_touch_effective`, `.dont_use_cells_effective`, `dont_touch_sources`, or `.read_only_effective` to make it easy to see exactly what objects were set in the library (see `-computed` to include them).

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `report_preserves` parameter. For a detailed description of the command and all its parameters, use the man command: `man report_preserves`
`-computed`	Includes computed inheritance attributes such as `.dont_touch_effective`, `.dont_touch_sources`, `.read_only_effective`, and `.dont_use_cells_effective` attributes in the report (unless filtered by `-dont_touch` or `-dont_use`). This causes all objects to inherit a non-default value from a parent or the library to appear in the report. *Data_type*: bool, optional
`-dont_touch`	Displays the report only related to `dont_touch`: `.dont_touch*`, `.read_only*`, and `is_ilm`. *Data_type*: bool, optional
`-dont_use`	Displays the report only related to `dont_use`: `.dont_use_cells*` and `.use_cells*`. *Data_type*: bool, optional
`-obj_type {design | hinst | module | inst | net | hnet | pin | hpin | base_cell}`	Displays the report only on specific object types. *Data_type*: enum, optional

Examples

Example 1

```
set_db hinst:TDSP_CORE_INST .size_only true
```

Queries all hinsts

```
report_preserve -obj_type hinst -dont_touch
```

```
############################################################
# Generated by: Cadence Innovus 16.10-d085_1
# OS: Linux x86_64(Host ID vlsj-daven)
# Generated on: Thu Feb 18 15:22:28 2016
# Design: top
# Command: report_preserves
# -obj_type hinst
# -dont_touch
############################################################

hinsts
--------------------------------------------------------------------------------

hinst:TDSP_CORE_INST
dont_touch size_ok

Total of 1 hinst constrained

Query all objects
```

```
report_preserves

report_preserve -obj_type hinst -dont_touch

#################################################################
# Generated by: Cadence Innovus 16.10-d085_1
# OS: Linux x86_64(Host ID vlsj-daven)
# Generated on: Thu Feb 18 15:22:28 2016
# Design: top
# Command: report_preserves
#################################################################

base_cells
-------------------------------------------------------------------------------

base_cell:rom_512x16A
dont_use true
base_cell:ram_256x16A
dont_use true
base_cell:pllclk
dont_use true
base_cell:DCAP4BW
dont_touch true
dont_use true

Total of 4 base_cells constrained

hinsts
-------------------------------------------------------------------------------

hinst:TDSP_CORE_INST
dont_touch size_ok

Total of 1 hinst constrained

modules
-------------------------------------------------------------------------------

module:tdsp_core
dont_touch size_ok

Total of 1 module constrained
```

Note: Object types with no objects constrained should not be reported and unconstrained objects should not be reported.

Example 2

```
set_module_view -partition tdsp_core -type read_only
set_db hinst:TDSP_CORE_INST/MPY_32_INST .size_only true
```

Queries all hinsts and modules

```
report_preserve -obj_type hinst -dont_touch

#################################################################
# Generated by: Cadence Innovus 16.10-d085_1
# OS: Linux x86_64(Host ID vlsj-daven)
# Generated on: Thu Feb 18 15:22:28 2016
# Design: top
# Command: report_preserves
# -obj_type {hinst module}
# -dont_touch
#################################################################

hinsts
-------------------------------------------------------------------------------
hinst:TDSP_CORE_INST
read_only true

hinst:TDSP_CORE_INST/MPY_32_INST
dont_touch size_ok

Total of 2 hinst constrained

modules
-------------------------------------------------------------------------------
module:tdsp_core
read_only true
```

```
Total of 1 module constrained

report_preserve -obj_type hinst -dont_touch -computed

############################################################
# Generated by: Cadence Innovus 16.10-d085_1
# OS: Linux x86_64(Host ID vlsj-daven)
# Generated on: Thu Feb 18 15:22:28 2016
# Design: top
# Command: report_preserves -obj_type hinst -dont_touch -computed
############################################################

hinsts
------------------------------------------------------------
hinst:TDSP_CORE_INST
dont_touch_effective true
read_only_effective true
read_only true

hinst:TDSP_CORE_INST/MPY_32_INST
dont_touch_effective size_ok
dont_touch_sources {user size_ok} {parent none} {read_only_effective none}
read_only_effective true

hinst:TDSP_CORE_INST/MPY_32_INST/addinc_add_62_54_8
dont_touch_effective true
dont_touch_sources {user false} {parent size_ok} {read_only_effective true}
dont_touch size_ok
read_only_effective true
```

3 hinsts constrained.

report_rc

```
report_rc
[-help]
out_file
[-inst string]
[-layer string]
[-max_second_rc float]
```

Calculates the first and second RC values. The first RC value is the longest continuous vertical `rx` cut for a cell, where:

- Rx cut is always in the left or right boundaries of the cell, and at times in the internal cell.
- It is the longest vertical bar (number of rows) in which the cell's `rx` cut participate in.
- Multiple-height cells contribute the rows of height.

The second RC value is the largest distance (in microns) from the left or right boundary of the cell to the second `rx` cut in the left and right directions.

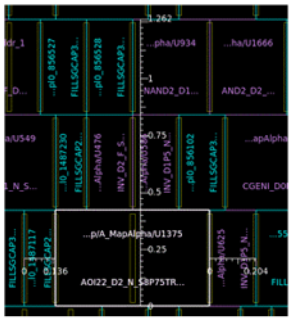
Parameters

`-help`	Prints a brief description that includes the type and default information for each `report_rc` parameter. For a detailed description of the command and all its parameters, use the man command: `man report_rc`
`out_file`	Specifies report filename. *Data_type*: string, required
`-inst` *`string`*	Specifies the distance from inst to report. *Data_type*: string, optional
`-layer` *`string`*	Specifies the layer of obs to calculate. *Data_type*: string, required
`-max_second_rc` *`float`*	Specifies the maximum distance to calculate the second RC. *Default*: `2` *Data_type*: float, optional

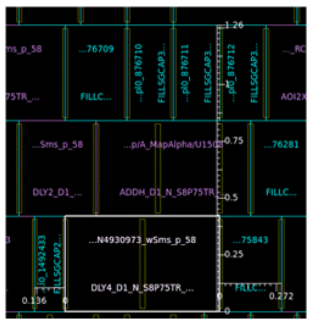
Examples

Refer to the following examples:

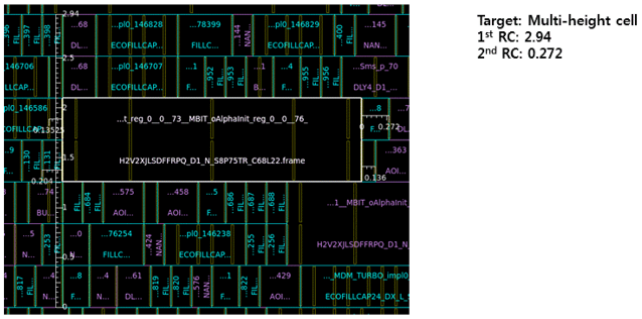
Example 1: STD Cell 1



Example 2: STD Cell 2



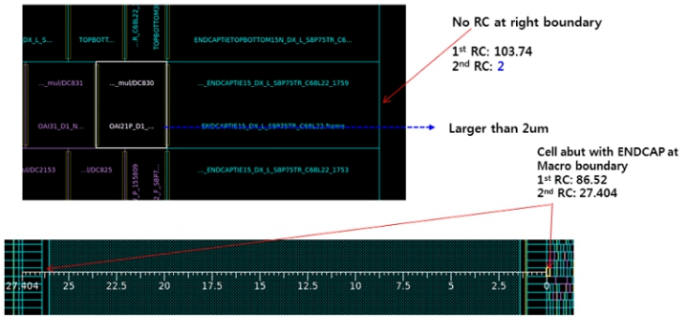
Example 3: Special Cells (Multi-height)



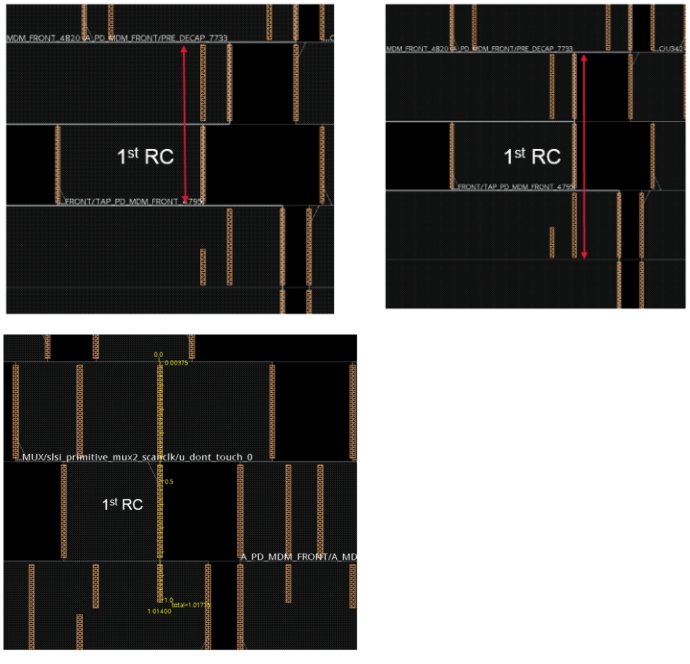
Example 4: Special Cells (ENDCAP)

Note that:

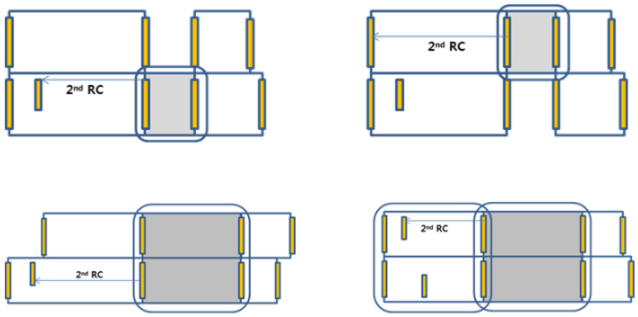
- An ENDCAP cell has a single RC.
- The cell with ENDCAP must have a long second RC.
- The second RC effect is saturated at 2um. It means that the second RC value for the cell is 2 um.



Example 5: First RC Values in Various Scenarios



Example 6: Second RC Values in Various Scenarios



Example 7: Input/Output Format of First and Second RC Generation

```
derive_rxcuts {
    -2nd_rc_saturation_length 2nd RC saturation length
    -output OUTPUT_FILE

}

innovus_shell> derive_rxcuts -2nd_rc_saturation_length  2.0 –output  ./rxcut.info.txt

dervie_rxcuts output format: {full_name  ref_name location  orientation  unit_cell_height  1st_RC 2nd_RC}
```

```
u_ee_toplevel/u_msg_u_rx_path/u_rx_conv_u_load_convert_top_LOAD_CONVERTER_LANES_1__u_load_convert_lane/DC2553
ln10lpp_sc_s8p75t_flk_rvt_c68l20_25c_sadhm_lvf_d:NAND2_D1_F_S8P75TR_C68L20.frame 65.09 392.28 MX 0.42 86.52 27.404
```

1st RC 2nd RC

report_via_def

```
report_via_def
[-help]
[-detail]
[-regular]
[-special]
```

Reports the single and multi-cut via summary for special and/or regular vias. Such a summary could be used for generating statistics regarding the usage of single and multi-cut used vias.

Parameters

`-help`	Prints a brief description that includes the type and default information for each `report_via_def` parameter. For a detailed description of the command and all its parameters, use the man command: `man report_via_def`
`-detail`	Prints the detailed report related to every via def. *Data_type*: bool, optional
`-regular`	Prints the report related to regular routing vias. *Data_type*: bool, optional
`-special`	Prints the report related to special routing vias. *Data_type*: bool, optional

Examples

Use the following command to display a detailed report related to every via:

```
report_via_def -detail
# Via-def                                Layer(cut)     Single-cut     Total
#--------------------------------------------------------------------------
# VIA1_20_7_20_7_VxaRECT_V                     VIA1       0            29733
# VIA1_20_7_20_7_VxaRECT_N                     VIA1       0            21322
# VIA1_20_7_20_7_VxaRECT_S                     VIA1       0            19468
# VIA1_20_7_7_20_VxaRECT_V                     VIA1       0             1266
# VIA1_0_25_0_25_VxaRECT_N49                   VIA1       0           152109
# VIA1_0_25_0_25_VxaRECT_S49                   VIA1       0           188660
...
#--------------------------------------------------------------------------
# Total                                                              10149027
```

Use the following command to display a report about regular routing vias, including single and multi-cut vias number (ratio):

```
report_via_def -regular
# Layer(cut)              Single-cut               Multi-cut          Total
#--------------------------------------------------------------------------
# VIA1              1872843 ( 74.2%)         649834 ( 25.8%)        2522677
# VIA2              1514819 ( 48.1%)        1632913 ( 51.9%)        3147732
# VIA3               914028 ( 50.4%)         900507 ( 49.6%)        1814535
# VIA4               514637 ( 47.1%)         578552 ( 52.9%)        1093189
# VIA5               203532 ( 40.2%)         302433 ( 59.8%)         505965
# VIA6                69496 ( 28.5%)         174355 ( 71.5%)         243851
```

```
# VIA7                11181 ( 13.9%)          69227 ( 86.1%)          80408
# VIA8                 4634 ( 10.1%)          41441 ( 89.9%)          46075
#-------------------------------------------------------------------------
# Total             5105170 ( 54.0%)        4349262 ( 46.0%)        9454432
```

report_wire_path

```
report_wire_path
[-help]
[-detail]
-end string
[-include_pin]
-start string
[-tcl [-no_output]]
```

Extracts the routing topology report in a point-to-point manner (for example between routing layer, wire length, via, resistor, and capacitance). The report contains the information such as timing path for each via, wire length, and width of path.

Parameters

`-help`	Prints a brief description that includes the type and default information for each `report_wire_path` parameter. For a detailed description of the command and all its parameters, use the man command: `man report_wire_path`
`-detail`	Prints the detailed information related to the path. *Data_type*: bool, optional
`-end string`	Specifies the end terminal name, which can be a `pin`, `port`, `hpin`, and `hport`. *Data_type*: string, required
`-include_pin`	Specifies whether to include pin in the report. *Data_type*: bool, optional
`-no_output`	Specifies whether to print the path when `-tcl` is used. *Data_type*: bool, optional
`-start string`	Specifies the start terminal name, which can be a `pin`, `port`, `hpin`, and `hport`. *Data_type*: string, required
`-tcl`	Specifies whether to return the Tcl dict of the wire path. *Data_type*: bool, optional

safety Category Attributes

The root attributes in the `safety` category are:

- safety_flow_enable
- safety_midas_enable
- safety_ser_cells
- safety_tmr_isolate_clock
- safety_tmr_spacing
- safety_tmr_spacing_x
- safety_tmr_spacing_y

To set the value of any root attribute, use the `set_db` command:

```
set_db attribute_name value
```

To get the current value of any root attribute, use the `get_db` command:

```
get_db attribute_name
```

To get the current value of all attributes in the `tech` category, use the following command:

```
get_db -category safety
```

safety_flow_enable

`safety_flow_enable {true | false}`

Default: `false`

Data_type: bool, read/write

Specifies the master switch to enable the implementation and analysis of the safety-mechanisms defined in the USF file.

safety_midas_enable

`safety_midas_enable {none |` fmeda `|` physical`}`

Default: `none`

Data_type: string, read/write

Enables the Unified Safety Format (USF) technology. The supported values of this attribute are:

- `none`: Does not enable this attribute.
- `fmeda`: Enables the USF command set for the failure mode effect and diagnostic analysis.
  Note: This value is supported only in Genus.
- `physical`: Enables the USF command set for physical implementation of safety mechanisms.
  Note: This value is supported in both Innovus and Genus.

To enable USF in Innovus, you must set `safety_midas_enable` to `physical`.

```
set_db safety_midas_enable physical
```

You will see a message like this:

```
Cadence Midas USF - Physical
Copyright 2022 Cadence Design Systems, Inc. All rights reserved worldwide.
Cadence and the Cadence logo are registered trademarks of Cadence Design Systems Inc.
in the United States and other countries.
Midas USF - Physical Version: 22.03.001
physical
```

ⓘ Once USF Physical is loaded, it is not possible to disable or reload it in the current session.

safety_ser_cells

`safety_ser_cells *string*`

Default: `""`

Data_type: string, read-only

Specifies the list of `base_cells` allowed for those insts, which have a Soft Error Resilience (SER) safety mechanism.

safety_tmr_isolate_clock

`safety_tmr_isolate_clock {none | unique_driver}`

Default: `none`

Data_type: enum, read/write

Specifies the type of clock isolation required for Triple Modular Redundancy (TMR) flops. When set to `unique_driver`, the clock pin of every TMR flop is directly driven by its own unique clock cell with `fanout = 1`.

safety_tmr_spacing

`safety_tmr_spacing *microns*`

Default: `0.0`

Data_type: coord, read/write

Specifies the radial spacing in any direction that should be maintained between the three flops in a TMR group. The value of this attribute is measured in microns.

Note: The `safety_tmr_spacing_x`, `safety_tmr_spacing_y`, and `safety_tmr_spacing` attributes are mutually exclusive.

safety_tmr_spacing_x

`safety_tmr_spacing_x *microns*`

Default: `0.0`

Data_type: coord, read/write

Specifies the horizontal spacing that should be maintained between the three flops in a TMR group. The value of this attribute is measured in microns. Only one value of either `safety_tmr_spacing_x` or `safety_tmr_spacing_y` needs to be met.

Note: The `safety_tmr_spacing_x`, `safety_tmr_spacing_y`, and `safety_tmr_spacing` attributes are mutually exclusive.

safety_tmr_spacing_y

`safety_tmr_spacing_y *microns*`

Default: `0.0`

Data_type: coord, read/write

Specifies the vertical spacing that should be maintained between the three flops in a TMR group. The value of this attribute is measured in microns. Only one value of either `safety_tmr_spacing_x` or `safety_tmr_spacing_y` needs to be met.

Note: The `safety_tmr_spacing_x`, `safety_tmr_spacing_y`, and `safety_tmr_spacing` attributes are mutually exclusive.

tech Category Attributes

The root attributes in the `tech` category are:

- tech_db_units
- tech_finfet_grid_direction
- tech_finfet_grid_offset
- tech_finfet_grid_pitch

- tech_inst_mask_shift_layers
- tech_mfg_grid

To set the value of any root attribute, use the `set_db` command:

```
set_db attribute_name value
```

To get the current value of any root attribute, use the `get_db` command:

```
get_db attribute_name
```

To get the current value of all attributes in the `tech` category, use the following command:

```
get_db -category tech
```

tech_db_units

`tech_db_units int`

Default: `{}`

Data_type: int, read-only

Specifies the database units per micron from LEF UNITS statement or OA techfile, unless `init_min_dbu_per_micron` is larger and overrides the LEF/OA value.

tech_finfet_grid_direction

`tech_finfet_grid_direction enum`

Default: `vertical`

Data_type: enum {vertical horizontal}, read/write

Specifies an enum value indicating the direction of the finfet grid from the LEF FINFET statement or OA.

tech_finfet_grid_offset

`tech_finfet_grid_offset coord`

Default: `{}`

Data_type: coord, read/write

Specifies the finfet grid offset from `0` in microns from the LEF FINFET statement or OA.

tech_finfet_grid_pitch

`tech_finfet_grid_pitch coord`

Default: `{}`

Data_type: coord, read/write

Specifies the finfet grid pitch in microns from the LEF FINFET statement or OA.

tech_inst_mask_shift_layers

`tech_inst_mask_shift_layers layer*`

Default: `{}`

Data_type: layer*, read-only

Specifies the ordered list of layer pointers to the layers that allow instance mask shifting. This is the list of layers that show up in the DEF COMPONENTMASK statement for layers that can have mask values shifted in standard cells.

tech_mfg_grid

`tech_mfg_grid coord`

Default: `{}`

Data_type: coord, read-only

Specifies the manufacturing grid from LEF MANUFACTURINGGRID statement or OA techfile.

Note: This is the default value for each layer, but it can be overridden for specific layers. In some technologies, the `mfg_grid` for higher metal layers is coarser than for the lower layers.

uniquify

```
uniquify
[-help]
{[design | module] | -new_master hinsts} [-exclude modules]
[-verbose]
```

Supports regrouping of master-clone partitions inside the non-unique modules. This command uniquifies or regroups the master/clones (one master module used by two or more `hinst` clones) by creating unique modules for each `hinst` clone, as needed. In all the cases, the new module names are generated by appending the '`_number`' suffix (such as `_1` and `_2`) sequentially to the original module name for creating a unique module name.

Parameters

`-help`	Prints a brief description that includes the type and default information for each `uniquify` parameter. For a detailed description of the command and all its parameters, use the man command: `man uniquify`
`design` `|` *`module`*	Regroups the master/clones within the *`design`* or *`module`* and recursively below it. This assures that all the `hinsts` inside those would have their own unique module, except for the modules specified using the `-exclude` parameter. In case multiple `hinsts` are referenced to the specified `module`, those will remain as clones. *Data_type*: (design|module), required
`-exclude` *`modules`*	Excludes the specified modules from unification or regrouping. Use this parameter if a specific sub-module is a partition, which will be implemented and shared only once. *Data_type*: (module)+, optional
`-new_master` *`hinsts`*	Splits the `hinst` clones of a master module into the clones of a new master. The new master is also regrouped, except for the modules specified using the `-exclude` parameter. It may also happen that one of such `hinst` clones is forced to have a new master (refer to the **Examples** section). In case a `hinst` is not a clone of the same module, an error message is displayed. *Data_type*: (hinst)+, required
`-verbose`	Reports the `hinst` clones (if any), which are regrouped. *Data_type*: bool, optional

Examples

- The following command uniquifies the whole design, where the `h1`, `h2`, and `h3 hinsts` are clones of the `mod1` module, and the `h4` and `h5 hinsts` (clones of the `mod2` module) are inside `mod1`:
```
uniquify top -verbose
#hinst name module name
h1 mod1 (unchanged)
h2 mod1_1
h3 mod1_2
h1/h4 mod2 (unchanged)
h1/h5 mod2_1
h2/h4 mod2_2
h2/h5 mod2_3
h3/h4 mod2_4
h3/h5 mod2_5
```
  Note that the first clone of each module is not modified. So, `h1` still uses `mod1`, and `h1`/`h4` still use `mod2`. However, these are still reported to see the `hinst` clone, which forces a new module for the other `hinst` clones.
- Using the same scenario as in the previous example, the following command uniquifes the design below `mod1`:
```
uniquify mod1 -verbose
#hinst name module name
h1/h4 mod2 (unchanged)
h1/h5 mod2_1
h2/h4 mod2 (unchanged)
h2/h5 mod2_1
h3/h4 mod2 (unchanged)
h3/h5 mod2_1
```
  Note that only the design inside `mod1` is uniquified. So, `h1`, `h2`, and `h3` are still the clones of `mod1`, and there are still master/clones grouping between `h1`, `h2`, and `h3`. For instance, all the `h4`s use `mod2`, and all the `h5`s use `mod2_1`.
- Using the same scenario as in the first example, the following command splits `h1` and `h2` into a new master. This is normally done, if `h1`, `h2`, and `h3` are partitions and need a different implementation than `h3`. It is recommended not to uniquify `mod2`, if it is a lower-level partition, which will be implemented once and shared by all `h1`, `h2`, and `h3`.
```
uniquify -new_master {h1 h2} -exclude mod2 -verbose
#hinst name module name
h1 mod1_1
h2 mod1_1
```
  Note that `h3` is still using `mod1`, and the `mod2 hinsts` inside `mod1` and `mod1_1` (such as `h4` and `h5`) are still the clones of `mod2`. If there is also a `mod3` inside `mod1`, it will also be uniquified, but nothing inside `mod2` is uniquified.
- Using the same scenario as in the first example, the following command splits `h1` and `h4` into a new master. This implicitly forces `h1` to be uniquified from `h2` and `h3`.
```
uniquify -new_master {h1/h4} -verbose
#hinst name module name
h1 mod1_1 (forced by lower level change)
h1/h4 mod2_1
```
  If there is a `mod4` inside `mod2_1`, it will also be uniquified, but it will not be shared with `mod2`.

4

Bus Plan Commands

- create_bus_guide
- delete_bus_guides
- gui_clear_bus_guide_colors
- gui_show_bus_guide_multi_colors
- update_bus_guide

create_bus_guide

```
create_bus_guide
[-help]
[-type {hard| soft}]
-net_group netGrpName
-layer {id | id1:id2}
{[{-center_line {x1 y1 x2 y2} -width value} [-begin_extension value] [-end_extension
value]] | -rect {x1 y1 x2 y2}}
```

Creates bus guides associated with net groups, to guide routing for all the nets of the specified net group. You can create horizontal, vertical, and 45-degree bus guide segments. 45-degree bus guides are required for flip chip routing.

Use this command after defining a net group that will be associated with the new bus guide segment.

After a bus guide is created, you can modify its layer or width later by using the `update_bus_guide` command.

Parameters

`-help`	Prints a brief description that includes type and default information for each `create_bus_guide` parameter. For a detailed description of the command and all of its parameters, use the man command: `man create_bus_guide`.
`-begin_extension` *value*	Specifies the value, in microns, by which the bus guide segment is to be extended in the beginning. *Data_type*: float, optional
`-center_line` *x1 y1 x2 y2*	
	Specifies the coordinates of the center line of the bus guide segment. The direction of the guide segment is derived from the `x1 y1 x2 y2` coordinate values. **Note:** Non-orthogonal bus guide segments are not supported. *Data_type*: rect, required
`-end_extension` *value*	Specifies the value, in microns, by which the bus guide segment is to be extended in the end. *Data_type*: float, optional
`-layer {`*id* | *id1*`:`*id2*`}`	Specifies the routing layer constraint honored by the router. ⓘ Routing done on non-preferred routing layers or on layers which are not part of the specified layer constraint can have high cost implications *Data_type*: string, required
`-net_group` *netGrpName*	Specifies the name of the net group that will be associated with the new bus guide segment that you are creating. *Data_type*: string, required
`-rect {`*x1 y1 x2 y2*`}`	Specifies the lower-left and upper-right coordinates of the rectangle formed by the bus guide segment. *Data_type*: rect, required

`-type {hard |soft}`	Specifies the constraint type of bus guide: • `hard` - After running Early Global Route (eGR), NanoRoute (NR), or NanoRoute High Frequency Router (NRHF), the route obeys the path of the bus guide. • `soft` - After running eGR, NR, or NRHF, the bus guide only guides the route path. The tool can route the net out of the bus guide. *Default*: `soft` (If you don't specify the `-type` parameter to define the constraint type, the value of constraint type is taken as `soft`.) *Data_type*: enum, optional
`-width` *value*	Specifies the width, in microns, of the bus guide segment that you are creating. *Data_type*: float, optional

Example

- The following command creates a horizontal bus guide segment whose width is `120` microns, and layer constraint is from METAL4 to METAL8:
  ```
  create_bus_guide -net_group netGroup1 -center_line 4421.8 10749.36 4960.8 10749.36 -width 120 -layer METAL4:METAL8
  ```
- The following command creates a bus guide segment whose lower-left coordinate is at pt {`464 1240`} and upper-right coordinate is at pt {`528 1260`} for the net group `test` and with a layer constraint of Metal1 to Metal4:
  ```
  create_bus_guide -net_group test -layer {1:4} -rect {464 1240 528 1260}
  ```
- The following command creates a 45-degree bus guide segment:
  ```
  create_bus_guide -net_group test3Group -center_line 400 500 450 550 -width 20 -layer {METAL7:METAL8}
  ```

Related Information

- `create_net_group`
- `delete_bus_guides`
- `update_bus_guide`
- Bus Planning chapter in the *User Guide*
  - *Creating a Bus Guide*
- Edit Menu chapter in the *Menu Reference*
  - *Bus Guide*

delete_bus_guides

```
delete_bus_guides
[-help]
[-area x1 y1 x2 y2]
[-net_groups {netGroup | {list_of_net_groups}}]
[-layers {id | id1:id2}]
[-direction {h | v}]
[-all]
```

Deletes a bus guide.

If you do not specify any option, by default, the command delete all the bus guide segments.

Use this command after creating a bus guide segment.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `delete_bus_guides` parameter. For a detailed description of the command and all of its parameters, use the man command: `man delete_bus_guides`.
`-all`	Deletes all bus guide segments in the design. *Data_type*: bool, optional
`-area` *x1 y1 x2 y2*	Specifies the coordinates of the area from which the bus guide segments are deleted. *Data_type*: rect, optional
`-direction {h | v}`	Specifies the direction of the bus guide segment to delete. This can be `h` (horizontal) or `v` (vertical). *Data_type*: enum, optional
`-layers {`*id* `|` *id1*`:`*id2*`}`	Specifies the layer or layer range from which bus guide segments are deleted. The command deletes bus guide segments with the exact specified layer information and not a subset of it. For example, for a bus guide segment with layer constraint `2:5` if you specify `-layers 5`, the segment is not deleted. To delete this guide segment you need to specify `-layers 2:5` completely. *Data_type*: string, optional
`-net_groups {`*netGroup*`|` `{`*list_of_net_groups*`}}`	Specifies the name of the net group or list of net groups from which the bus guide segments are deleted. You can also use wildcard (* or ?) for specifying the net group name. *Data_type*: string, optional

Example

The following command deletes all the bus guide segments associated with the net group `abcBus` that has layer constraint from `Metal 3` to `Metal 7`:

```
delete_bus_guides -net_groups abcBus -layers 3:7
```

Related Information

- `create_bus_guide`
- `update_bus_guide`
- Bus Planning chapter in the *User Guide*
  - *Creating a Bus Guide*
- Edit Menu chapter in the *Menu Reference*
  - *Bus Guide*

gui_clear_bus_guide_colors

`gui_clear_bus_guide_colors`

Clears the highlighted bus guide colors.

You can use this command to clear colors after `gui_show_bus_guide_multi_colors` has been used to color the bus guide(s).

Parameters

`-help`	Prints out the command usage. For a detailed description of the command and all of its parameters, use the man command: `man gui_show_bus_guide_multi_colors`

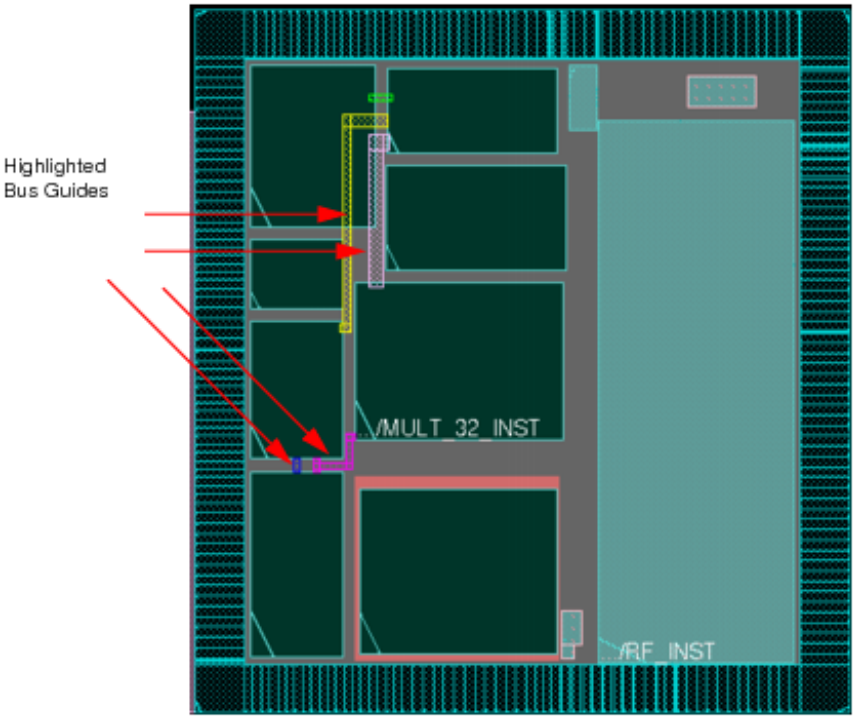
Related Information

- `gui_show_bus_guide_multi_colors`
- Bus Planning chapter in the *User Guide*
  - *Highlighting and Dehighlighting the Bus Guide*
- Edit Menu chapter in the *Menu Reference*
  - *Bus Guide*

gui_show_bus_guide_multi_colors

`gui_show_bus_guide_multi_colors`

Colors all the bus guides. Use this command to make it easy to distinguish between different bus guide objects in the design.



Specify `gui_clear_bus_guide_colors` to clear the bus guide colors.

Parameters

`-help`	Prints out the command usage. For a detailed description of the command and all of its parameters, use the man command: `man gui_clear_bus_guide_colors`

Related Information

- `gui_clear_bus_guide_colors`
- Bus Planning chapter in the *User Guide*
  - *Highlighting and Dehighlighting the Bus Guide*
- Edit Menu chapter in the *Menu Reference*
  - *Bus Guide*

update_bus_guide

```
update_bus_guide
[-help]
[-avoid_obstacles]
[-horizontal_layers {id | id1:id2}]
[-no_snap]
[-resize_direction {top_right bottom_left}]
[-type {hard | soft}]
[-vertical_layers {id | id1:id2}]
[-width value | -net_group name | {{-route_rule rulename | {-wire_width value -
wire_spacing value}} [-with_shield ]}]
```

Updates the selected bus guide as specified. The bus guide can expand automatically with pre-route or routing blockage.

Use the `update_bus_guide` command when you need to modify the layer or width information of an existing bus guide. Before using the `update_bus_guide` command, select the bus guide with the `select_obj` command.

Parameters

`-help`	Prints a brief description that includes type and default information for each `update_bus_guide` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man update_bus_guide`
`-avoid_obstacles`	Specifies whether to avoid pre-routes and blockages (detour). If this option is not specified, the bus guide is expanded with the pre-route and routing blockage. *Data_type*: bool, optional
`-horizontal_layers {id | id1:id2}`	Specifies the horizontal layer *id* or a layer range. Use this parameter to change the horizontal layer of the selected bus guide. *Data_type*: string, optional

`-net_group` *name*	Specifies the net group name. You must specify an existing net group. *Data_type*: (net_group), optional
`-no_snap`	Specifies whether or not to snap the bus guide. *Data_type*: bool, optional
`-resize_direction {top_right bottom_left}`	Specifies the resize direction as follows: • `top_right`: Expands the selected bus guide from the top or right edge. • `bottom_left`: Expands the selected bus guide from the bottom or left edge. If this option is not specified, the bus guide is expanded from the center line. *Data_type*: enum, optional
`-route_rule` *rulename*	Specifies the rule name or default. Use this parameter to change the selected bus guide width as per the wire width or wire spacing defined in the specified LEF rule. If *rulename* is not specified, default LEF rule is used. **Note**: This parameter cannot be used with the `-width`, `-wire_width`, or `wire_spacing` parameters. *Data_type*: string, optional
`-type {hard | soft}`	Specifies the new constraint type of the bus guide: • `hard` - After running Early Global Route (eGR), NanoRoute (NR), or NanoRoute High Frequency Router (NRHF), the route obeys the path of the bus guide. • `soft` - After running eGR, NR, or NRHF, the bus guide only guides the route path. The tool can route the net out of the bus guide. *Data_type*: enum, optional
`-vertical_layers {`*id* `|` *id1:id2*`}`	Specifies the vertical layer *id* or a layer range. Use this parameter to change the vertical layer of the selected bus guide. *Data_type*: string, optional

`-width value`	Specifies bus guide width. Use this parameter to change the selected bus guide width to the specified value. **Note**: This parameter cannot be used with the `-route_rule`, `-wire_width`, or `wire_spacing` parameters. *Data_type*: float, optional
`-wire_spacing value`	Specifies wire spacing. Use this parameter to change the selected bus guide width as per the specified wire spacing value. **Note**: This parameter cannot be used with either the `-route_rule` parameter or the `-width` parameter. *Data_type*: float, optional
`-wire_width value`	Specifies wire width. Use this parameter to change the selected bus guide width as per the specified wire width value. **Note**: This parameter cannot be used with either the `-route_rule` parameter or the `-width` parameter. *Data_type*: float, optional
`-with_shield`	Adds shield wire base on the shield net property defined by the following root attributes in the `edit_wire` category: `edit_wire_outer_shield_spacing`, `edit_wire_outer_shield_width`, and `edit_wire_shield`. When specified with: • `-wire_width` and `-wire_spacing`, the tool considers only the outer shielding nets. • `-route_rule`, the tool considers the shielding nets between wires. *Data_type*: bool, optional

Example

- The following command updates the width of the selected bus guide as per the NDR *doublewidth* defined in LEF. It also changes the horizontal layer of the selected bus guide to 3 and considers the shielding nets between wires.
```
update_bus_guide -route_rule doublewidth -horizontal_layers 3 -with_shield
```

Related Information

- `create_bus_guide`
- `delete_bus_guides`
- `edit_wire` category attributes
- Bus Planning chapter in the *User Guide*
- Edit Menu chapter in the *Menu Reference*
    - *Bus Guide*

5

Check Commands and Attributes

- check_ac_limit Category Attributes
- check_ac_limits
- check_antenna
- check_connectivity
- check_cut_density
- check_drc
- check_drc Category Attributes
- check_endcaps
- check_io_to_bump_connectivity
- check_isolated_cuts
- check_litho
- check_metal_density
- check_pg_shorts
- check_power_vias
- check_process_antenna
- check_tieoffs
- check_well_antenna
- check_well_taps
- check_wire_gaps
- create_marker
- delete_drc_markers
- delete_markers
- get_signoff_verify_design_config
- gui_show_markers
- opt_pattern
- read_drc_markers
- read_markers
- report_markers
- set_isolated_cut_rule
- set_signoff_verify_design_config
- set_snap_grid
- signoff_verify_design
- write_drc_markers

check_ac_limit Category Attributes

The root attributes in the `check_ac_limit` category are:

- check_ac_limit_additional_individual_violation_report
- check_ac_limit_avg_recovery

- check_ac_limit_current_file
- check_ac_limit_current_scale_factor
- check_ac_limit_current_scale_table
- check_ac_limit_default_freq_for_unconstrained_nets
- check_ac_limit_delta_temperature
- check_ac_limit_delta_temperature_layer_list
- check_ac_limit_detailed
- check_ac_limit_effort_level
- check_ac_limit_em_cdf_percentage
- check_ac_limit_em_limit_scale_factor
- check_ac_limit_em_limit_scale_table
- check_ac_limit_em_res_width
- check_ac_limit_em_temperature
- check_ac_limit_em_temperature_layer_list
- check_ac_limit_em_threshold
- check_ac_limit_enable_seb
- check_ac_limit_env_temperature
- check_ac_limit_extraction_tech_file
- check_ac_limit_force_hold_view
- check_ac_limit_handle_pin_obs_via
- check_ac_limit_ict_em_models
- check_ac_limit_ircx_models
- check_ac_limit_lifetime
- check_ac_limit_max_error
- check_ac_limit_method
- check_ac_limit_min_peak_duty_ratio
- check_ac_limit_min_peak_freq
- check_ac_limit_net_file
- check_ac_limit_nets
- check_ac_limit_out_file
- check_ac_limit_peak_td_method
- check_ac_limit_seb_lifetime
- check_ac_limit_seb_table
- check_ac_limit_seb_temperature
- check_ac_limit_selected
- check_ac_limit_simulation_net
- check_ac_limit_simulation_net_file
- check_ac_limit_skip_category_mode
- check_ac_limit_skip_net
- check_ac_limit_skip_net_file
- check_ac_limit_toggle
- check_ac_limit_use_db_freq
- check_ac_limit_use_qrc_tech

- check_ac_limit_use_rms_delta_t
- check_ac_limit_view

To set the value of any root attribute, use the `set_db` command:

```
set_db attribute_name value
```

To get the current value of any root attribute, use the `get_db` command:

```
get_db attribute_name
```

To get the current value of all attributes in the `check_ac_limit` category, use the following command:

```
get_db -category check_ac_limit
```

check_ac_limit_additional_individual_violation_report

`check_ac_limit_additional_individual_violation_report {true | false}`

Default: `false`

Data_type: bool, read/write

Specifies to generate separate violation reports for each current waveform calculation method (`rms`, `peak`, or `avg`). If this attribute is not specified, a single violation report file is generated using the `check_ac_limit_method {RMS AVG peak}` attribute. This attribute gives you the flexibility of generating either a single collated report or individual reports for each analysis method.

Example:

```
set_db  check_ac_limit_method {rms peak avg} \
check_ac_limit_out_file context1.rpt \
...
check_ac_limit_additional_individual_violation_report true
```

This command will generate 3 different reports, namely `context1.rpt_avg`, `context1.rpt_peak`, and `context1.rpt_rms`. If there are no EM violations for a method, the tool will not generate the corresponding separate report. In this example, if there are violations for `avg` and `rms` but none for `peak`, then it will generate only two reports, `context1.rpt_avg` and `context1.rpt_rms`.

check_ac_limit_avg_recovery

`check_ac_limit_avg_recovery em_recover`

Default: `0.0`

Data_type: double, read/write

Specifies the recovery factor used for calculating I_{avg} current density with recovery. By default, the recovery factor (per layer) is read from the Quantus techfile. If you specify `em_recover`, it overrides the Quantus tech `em_recover` factor for all layers used in I_{avg} limits. The minimum value of this attribute is `-1` and the maximum is `1`.

check_ac_limit_current_file

`check_ac_limit_current_file filename`

Default: `""`

Data_type: string, read/write

Specifies the current of a macro's output pins for EM calculation, in case the macro does not have the respective timing library. When the macro output pin current is set by the current file, the value calculated (if any) from AAE is overwritten.

check_ac_limit_current_scale_factor

`check_ac_limit_current_scale_factor {{avg value} {rms value} {peak value}}`

Default: `""`

Data_type: string, read/write

Scales the signal EM current by the respective factor for avg, rms, or peak current specified by `{avg value} {rms value} {peak value}`.

The value specified for this attribute must be > 0 and <= 10.

check_ac_limit_current_scale_table

`check_ac_limit_current_scale_table` *`current_scale_table_file`*

Default: `""`

Data_type: string, read/write

Specifies the name of the scale table file used for layer-based scaling of the signal EM current. The scale table contains the scale factor for avg, rms, or peak current for each layer.

check_ac_limit_default_freq_for_unconstrained_nets

`check_ac_limit_default_freq_for_unconstrained_nets` *`freq_in_Hz`*

Default: `0.0`

Data_type: double, read/write

Specifies the frequency for EM calculation when a net has no defined frequency or has a defined frequency of `0` Hz in the design.

check_ac_limit_delta_temperature

`check_ac_limit_delta_temperature` *`temp`*

Default: `5.0`

Data_type: double, read/write

Specifies the maximum temperature rise permitted in units of Celsius. This value is normally used in the Quantus tech file for the RMS current limit equation. For example:

```
em_jmax_ac_rms EQU sqrt(12.66 * delta_T * (w - 0.003)^2 * (w - 0.003 + 0.264) / (w - 0.003 + 0.0443))
```

check_ac_limit_delta_temperature_layer_list

`check_ac_limit_delta_temperature_layer_list {{`*`<LEF_layer_name> <delta_T_in_C>`*`}+}`

Default: `""`

Data_type: string, read/write

Specifies the delta temperature for specific LEF layers during signal EM RMS current limit analysis.

check_ac_limit_detailed

`check_ac_limit_detailed`

Default: `false`

Data_type: bool, read/write

Generates a detailed report containing information for all signal nets, including those without violations.

check_ac_limit_effort_level

`check_ac_limit_effort_level {low | medium | high}`

Default: `low`

Data_type: enum {low medium high}, read/write

Specifies the method used for current calculation of nets in the Signal ElectroMigration analysis flow. The two methods are:

- **non-simulation mode** - use the capacitance ratio to distribute the current down the stream from the driver to the receiver.
- **simulation mode** - simulates resistance and capacitance to get the true current on each segment. This mode is more accurate but requires a longer run time.

This attribute has the following three arguments:

- `low`: enables the non-simulation mode for all nets where the current distribution is based on the capacitance ratio.
- `medium`: enables the simulation mode for the nets with physical loops, and the non-simulation mode for the non-loop nets.

- `high`: enables the simulation mode for all nets.

The `medium` and `high` levels require a VTS-AA license.

Note: For the simulation mode with effort level `medium` or `high`:

- If you use `extract_rc`, run `set_db extract_rc_qrc_cmd_type partial extract_rc_qrc_cmd_file` *`qrc.user.cmd`*, where *`qrc.user.cmd`* must include the following CCL commands:
  ```
  extraction_setup -analysis em -cluster_vias_in_em true
  output_db -type spef -subtype "EXTENDED"
  ```

  Refer to the example below:
  ```
  output_db \
   -type spef -busbit_delimiter "[]" -escape_special_character false \
   -hierarchy_delimiter "/" -output_incomplete_nets true \
   -output_unrouted_nets true -pin_delimiter ":" -subtype "EXTENDED" \
   -disable_subnodes false -add_explicit_vias true \
   -include_parasitic_res_width true -include_parasitic_res_width_drawn true
  ```
- If you run standalone Quantus to generate the required SPEF file in the extended format separately, then instead of `extract_rc`, use the following command for signal EM analysis:
  `read_spef -keep_star_node_locations -extended` *`spef_file`*

check_ac_limit_em_cdf_percentage

`check_ac_limit_em_cdf_percentage` *`value`*

Default: `1e-9`

Data_type: bool, read/write

Calculates the Cumulative Distribution Factor (CDF) derating factor based on the EM rules specified in the ICT EM file, given by the `check_ac_limit_ict_em_models` attribute. This attribute is used to calculate the fail rate and reliability of a design. The value of the CDF percentage should be between 0 and 100.

When the CDF related rule exits in the ICT EM file, the software uses the default value `1e-09` for `check_ac_limit_em_cdf_percentage` if it is not specified.

check_ac_limit_em_limit_scale_factor

`check_ac_limit_em_limit_scale_factor {{avg` *`value`*`} {rms` *`value`*`} {peak` *`value`*`}}`

Default: `""`

Data_type: string, read/write

Scales the calculated signal EM limits by the respective factor for avg, rms or peak current limit specified by `{avg` *`value`*`} {rms` *`value`*`} {peak` *`value`*`}`. The software compares the current through the net against the scaled EM limit to report violations in signal EM.

The value specified for this attribute must be > 0 and <= 10.

check_ac_limit_em_limit_scale_table

`check_ac_limit_em_limit_scale_table` *`em_limit_scale_table_file`*

Default: `""`

Data_type: string, read/write

Specifies the name of the scale table file used for layer-based scaling of the signal EM limit. The scale table contains the scale factor for avg, rms, or peak current for each layer.

check_ac_limit_em_res_width

`check_ac_limit_em_res_width {drawn silicon}`

Default: `drawn`

Data_type: enum {drawn silicon}, read/write

Specifies width used to get EM limit.

check_ac_limit_em_temperature

```
check_ac_limit_em_temperature value
```

Default: `0.0`

Data_type: double, read/write

Specifies the temperature expected for the device and is used to look up the temperature scaling factor. By default, no scaling is done. The *value* is a float, in units of degrees Celsius.

check_ac_limit_em_temperature_layer_list

```
check_ac_limit_em_temperature_layer_list {{<LEF_layer_name> <em_temperature_in_C>}+}
```

Default: `""`

Data_type: string, read/write

Sets a different temperature for specific LEF layers during signal EM AVG current limit analysis.

check_ac_limit_em_threshold

```
check_ac_limit_em_threshold value
```

Default: `1.0`

Data_type: double, read/write

Specifies to report violations in signal EM above the threshold value. The value is the ratio of signal current (avg/peak/rms) to the respective EM limit.

check_ac_limit_enable_seb

```
check_ac_limit_enable_seb {true | false}
```

Default: `false`

Data_type: bool, read/write

Specifies to enable the Statistical ElectroMigration Budgeting (SEB) analysis.

check_ac_limit_env_temperature

```
check_ac_limit_env_temperature value
```

Default: `0.0`

Data_type: double, read/write

Specifies the ambient temperature used to calculate S_{DC} for SEB analysis.

check_ac_limit_extraction_tech_file

```
check_ac_limit_extraction_tech_file filename
```

Default: `""`

Data_type: string, read/write

Specifies an extraction technology file for the Signal ElectroMigration analysis flow. If this attribute is not specified, then by default the extraction technology file will be taken from the analysis view (current RC corner).

check_ac_limit_force_hold_view

```
check_ac_limit_force_hold_view
```

Default: `false`

Data_type: bool, read/write

Forces `set_signal_em_analysis_mode` to use the hold view for current calculation even during setup fixing.

check_ac_limit_handle_pin_obs_via

`check_ac_limit_handle_pin_obs_via`

Default: `false`

Data_type: bool, read/write

Specifies to consider the vias defined in the `OBS` (obstructions/blockages) section of LEF in the Signal ElectroMigration flow. By default, the software does not consider via in obstruction.

This attribute does not honor the via obstructions with "`SPACING 0`".

check_ac_limit_ict_em_models

`check_ac_limit_ict_em_models` *`file`*

Default: `""`

Data_type: string, read/write

Specifies the ICT-EM file which only contains the EM rules and excludes the RC extraction data in ICT. The software will use the EM rules in the specified ICT-EM file to calculate EM limits for signal net EM analysis even if there are embedded EM rules in the Quantus tech file.

check_ac_limit_ircx_models

`check_ac_limit_ircx_models {RC.ircx EMIR.ircx}|{RC.ircx}`

Default: `""`

Data_type: string

Specifies an encrypted or unencrypted RC/EM technology (.ircx) file. The foundry-specific IRCX file, which contains the process information for extraction and EM modeling, can be used when the ICT-EM file is not available for electromigration analysis.

You can either specify two IRCX files, one file containing the RC process data (`RC.ircx`) and the other file containing EM rules (`EM.ircx`), or a single IRCX file that includes both the RC and EM sections (`RC.ircx`). A single IRCX file is used for the mature nodes, and separate RC and EM IRCX files are used for the advanced process nodes.

check_ac_limit_lifetime

`check_ac_limit_lifetime` *`value`*

Default: `0.0`

Data_type: double, read/write

Specifies the total hours of operation expected for the device, and is used to look up the lifetime scaling factor. *`value`* is used to find the scaling factor by interpolating from a piece-wise-linear table in the LEF or the Quantus tech file (`em_jmax_ac model` with `jmax_lifetime` table). If there is no scaling factor table, no scaling is done. *`value`* is a float, in units of hours.

check_ac_limit_max_error

`check_ac_limit_max_error` *`value`*

Default: `10000`

Data_type: int, read/write

Controls the number of errors to be included in the signal EM violation report file. If you do not specify this attribute, the top 10000 signal nets with violation will be included in the violation report.

The `check_ac_limit_max_error` attribute can be used to control the size of the violation report file.

check_ac_limit_method

```
check_ac_limit_method {rms peak avg}
```

Default: ""

Data_type: string, read/write

Specifies the current waveform calculation method. You can specify one or more of the following options:

- `avg`: Checks I_{avg} limits and reports any average current limit violations in the design.
- `peak`: Checks I_{peak} limits and reports any peak current limit violations in the design.
- `rms`: Checks I_{rms} limits and reports any RMS current limit violations in the design.

check_ac_limit_min_peak_duty_ratio

```
check_ac_limit_min_peak_duty_ratio duty_ratio
```

Default: `0.0`

Data_type: double, read/write

Sets the minimum duty ratio value for I_{peak} calculation. If the calculated duty ratio of a net is lower than the specified value, the specified value will be used to calculate I_{peak}. The minimum value for *duty_ratio* is `0.0`and the maximum is `1.0`.

check_ac_limit_min_peak_freq

```
check_ac_limit_min_peak_freq freq_in_Hz
```

Default: `0.0`

Data_type: double, read/write

Ignores I_{peak} current limit calculation for signal nets that have an effective frequency below the specified value (in Hertz).

check_ac_limit_net_file

```
check_ac_limit_net_file list_file
```

Default: `""`

Data_type: string, read/write

Specifies the net from file to check the signalEM analysis. The format is one net name per line, with comments using # as the first character. For example:

```
#list of nets to skip
net_a
net_b
net_c
```

check_ac_limit_nets

```
check_ac_limit_nets {netNames}
```

Default: `""`

Data_type: string, read/write

Specifies whether to check named or selected nets. If you name only one net, you do not need the braces around the net name. If you specify more than one net, you must separate the net names with a space.

check_ac_limit_out_file

```
check_ac_limit_out_file filename
```

Default: `""`

Data_type: string, read/write

Specifies the report file for the violation data. The report file includes name of the net, I_{rms}, interpolated `ACCURRENTDENSITY` limit (I_{limit}), layer, width, X and Y location, rise and fall time, effective frequency for the net (F_{eff}), Cnet and Vdd for each net.

check_ac_limit_peak_td_method

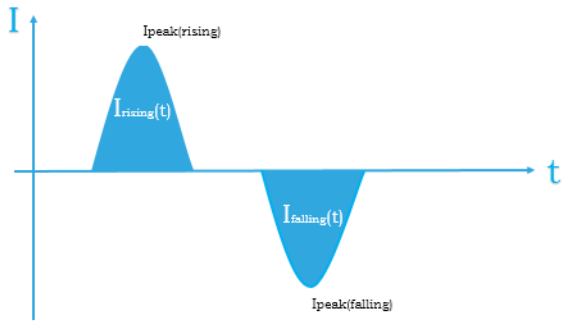
`check_ac_limit_peak_td_method {effective_width_from_integration effective_half_peak_width max_equivalent_dc_peak sum_half_peak_width}`

Default: `effective_width_from_integration`

Data_type: enum {effective_width_from_integration effective_half_peak_width max_equivalent_dc_peak sum_half_peak_width}, read/write

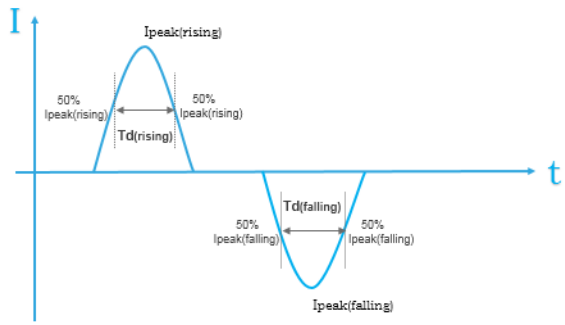
Specifies the Td calculation method for Ipeak check. The legal values are depicted below.

`effective_width_from_integration`



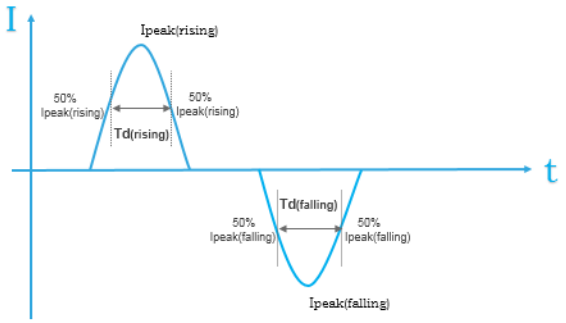
- $\mathrm{Td} = \frac{\int_0^{Tsw} |I(t)|dt}{\mathrm{Ipeak_measured}}$
  - Tsw = Period of one switching cycle (rising + falling)
  - Ipeak_measured = abs.max(Ipeak(rising), Ipeak(ralling))

`effective_half_peak_width`



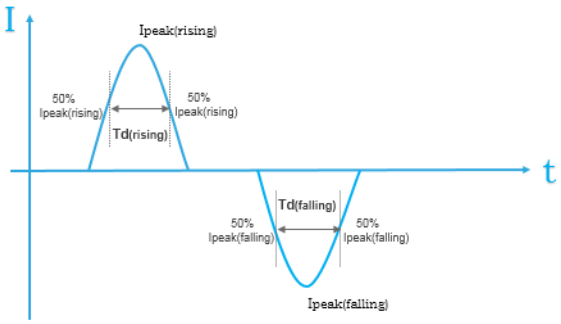
- When Ipeak(rising) >= Ipeak(falling)
  $\mathrm{Td} = \mathrm{Td(rising)} + (\frac{\mathrm{Ipeak(falling)}}{\mathrm{Ipeak(rising)}})^2 * \mathrm{Td(falling)}$
- When Ipeak(rising) < Ipeak(falling)
  $\mathrm{Td} = \mathrm{Td(falling)} + (\frac{\mathrm{Ipeak(rising)}}{\mathrm{Ipeak(falling)}})^2 * \mathrm{Td(rising)}$

max_equivalent_dc_peak



- $R(rising) = Td(rising)/(T/2)$
- $R(falling) = Td(falling)/(T/2)$
- Choose $\max(Ipeak_{(rising)}*sqrt(R_{(rising)}), Ipeak_{(falling)}*sqrt(R_{(falling)}))$ to compare against Ipeak_dc_limit
- $Ipeak_ac_limit = Ipeak_dc_limit/sqrt(R)$
- When $Ipeak_{(rising)}*sqrt(R_{(rising)}) >= Ipeak_{(falling)}*sqrt(R_{(falling)})$
  $Td = 2*Td(rising)$
- When $Ipeak_{(rising)}*sqrt(R_{(rising)}) < Ipeak_{(falling)}*sqrt(R_{(falling)})$
  $Td = 2*Td(falling)$

sum_half_peak_width



- $Td = Td_{(rising)} + Td_{(falling)}$

check_ac_limit_seb_lifetime

`check_ac_limit_seb_lifetime` *value*

Default: `43800` hours (that is, 5 years)

Data_type: double, read/write

Specifies the life time of product in hours.

check_ac_limit_seb_table

`check_ac_limit_seb_table` *table_filename*

Default: `""`

Data_type: string, read/write

Specifies the file name of the Failures in Time (FIT) table. This file is provided by the foundry. The file has the following columns:

- Layer
- MTF
- Sigma

- Activate energy
- Current density exponent

check_ac_limit_seb_temperature

`check_ac_limit_seb_temperature` *`temperature`*

Default: `""`

Data_type: double, read/write

Specifies the temperature used to update the median time to failure (MTF) with respect to the ambient temperature.

- If this temperature is specified, it would be used globally for all the resistors.
- If this temperature is not specified, the `seb_temperature` may take the following possible values depending on the specific circumstances:
  - If detailed delta temperature file is not used for SEB, the `seb_temperature` always takes the same value as that used for EM analysis.
  - If detailed delta temperature file is used for SEB, the `seb_temperature` is calculated by the following equation:
    ```
    seb_temperature  = env_temperature + delta_T
    ```
    Where `delta_T` is the maximum temperature rise, including joule heat and coupling heat from FEOL, in the resistor's bounding box with zero extension.

check_ac_limit_selected

`check_ac_limit_selected` *`filename`*

Default: `false`

Data_type: bool, read/write

Analyzes the net specified by selecting a net in the main GUI window.

check_ac_limit_simulation_net

`check_ac_limit_simulation_net` *`list_of_nets`*

Default: `""`

Data_type: string, read/write

Forces the specified nets to use the simulation mode.

check_ac_limit_simulation_net_file

`check_ac_limit_simulation_net_file` *`filename`*

Default: `""`

Data_type: string, read/write

Forces the nets in the specified file to use the simulation mode.

check_ac_limit_skip_category_mode

`check_ac_limit_skip_category_mode {0 | 1 | 2}`

Default: `1`

Data_type: int, read/write

Specifies to skip nets that belong to categories mapping to the specified mode. This parameter allows you to specify three modes, namely 0, 1 (default), and 2. The following table lists the common categories for all modes, and the categories specific to each mode:

Common Categories	0 Skip Category Mode	1 Skip Category Mode	2 Skip Category Mode
• Non top scope nets • Internal nets on boundary path • P/G nets • Constant nets • User-defined skipped nets • Nets below effective frequency threshold • Physically broken nets • Nets without loading capacitance annotation • Nets with SPEF back-annotation failures during simulation • Nets failed in analysis	• Slave nets by Verilog Assign • Nets (non-constant) without frequency annotation • Nets without physical shape for output pin • Nets without wire shape connection to output pin • Nets with zero pin current	• Equivalent nets by Verilog Assign • Nets without driver • Nets without receiver	• Nets without pins • Equivalent nets by Verilog Assign • Nets without driver • Nets without receiver

For example, if you set `set_signal_em_analysis_mode -skip_category_mode to 1`, the report file lists all categories listed in the *Common Categories* and *1 Skip Category Mode* columns.

Following is a snippet of the report file for mode **1**:

```
Total Nets in Design:              33451
Total Selected Nets in Design:         4
Total Analyzed Nets:                   0
Total Skipped Nets:                    4
  Non top scope nets:                                    0
  Internal nets on boundary path:                        0
  Equivalent nets by Verilog Assign:                     0
  P/G nets:                                              0
  Constant nets:                                         2
  User-defined skipped nets:                             0
  Nets below effective frequency threshold:              0
  Nets without driver:                                   1
  Nets without receiver:                                 1
  Physically broken nets:                                0
  Nets without loading capacitance annotation:           0
  Nets with SPEF back-annotation failures during simulation:  0
  Nets failed in analysis:                               0
```

check_ac_limit_skip_net

`check_ac_limit_skip_net` *`net_name_list`*

Default: `""`

Data_type: string, read/write

Specifies the net to be skipped in the signalEM analysis.

Refer to the example below:

```
{net_a net_b net_c}
```

check_ac_limit_skip_net_file

`check_ac_limit_skip_net_file` *`list_file`*

Default: `""`

Data_type: string, read/write

Specifies the net file to be skipped in the signalEM analysis.

The format is one net name per line, with comments using # as the first char. For example:

```
#list of nets to skip
net_a
net_b
net_c
```

check_ac_limit_toggle

`check_ac_limit_toggle` *`value`*

Default: `1.0`

Data_type: double, read/write

Specifies the switching factor for signal nets. A value of `1.0` means that if a flip-flop is clocked by a 20 Mhz clock, it changes state on 100% of the clock cycles.

Therefore, the signal has 20 million transitions, which is half the number of transitions of the input clock. As a result, the final switching frequency of the signal net is 10 Mhz.

Value Range: `0.0` to `1.0`

check_ac_limit_use_db_freq

```
check_ac_limit_use_db_freq
```

Default: `false`

Data_type: bool, read/write

Uses the database frequency value as the effective frequency per net.

check_ac_limit_use_qrc_tech

```
check_ac_limit_use_qrc_tech
```

Default: `false`

Data_type: bool, read/write

Forces I_{rms} checks to use the Quantus tech file instead of the technology LEF file. If either I_{peak} or I_{avg} is also checked, all checks will use the Quantus tech file, including I_{rms}. This attribute is enabled by default.

Note: The `-method avg` and `-method peak` analysis will only support AC current density rules from Quantus tech file. If either `avg` or `peak` is also specified, then all checks will use the Quantus tech file equations, including RMS.

check_ac_limit_use_rms_delta_t

```
check_ac_limit_use_rms_delta_t {true | false}
```

Default: `false`

Data_type: bool, read/write

When this parameter is set to `true`, the `delta_T` used to calculate `seb_temperature` would be based on the RMS delta T (joule heat only).

If this parameter is set to `false`, the `delta_T` used to calculate `seb_temperature` would be based on the sum of self-heat delta T and RMS delta T.

check_ac_limit_view

```
check_ac_limit_view viewName
```

Default: `null`

Data_type: string, read/write

Specifies the name of the view.

Related Information

Verifying AC Limit section in the Identifying and Viewing Violations chapter of the *Innovus Stylus Common UI User Guide.*

check_ac_limits

```
check_ac_limits
[-help]
```

Checks for the following types of AC current violations on signal nets.

- Root-mean-square (RMS) current limit violations (I_{rms}violations)
- Peak current limit violations (I_{peak}violations)

In addition, `check_ac_limits` can be used for average current limit (I_{avg}) analysis.

AC current limit violations are sometimes also referred as wire self-heat violations. DRC manuals have these current limits to avoid over-heating a signal-nets

wire with too much AC current. To prevent wire self-heating or AC signal electromigration, signal interconnects should be analyzed for their AC current carrying capacity and measured against the AC current limits specified by foundry.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `check_ac_limits` parameter. For a detailed description of the command and all its parameters, use the `man` command: `man check_ac_limits`

Related Information

Verifying AC Limit section in the Identifying and Viewing Violations chapter of the *Innovus Stylus Common UI User Guide.*

check_antenna

```
check_antenna
[-help]
[-detailed]
[-error integer]
[-nets {net_names} | -selected]
[-out_file file_name]
[-use {all | signal | power | clock}]
```

Verifies the process antenna effect (PAE) in multi-thread mode with geometry-based check through the advanced NanoRoute antenna checker feature. It also supports advanced antenna rules, such as `LEF58_ANTENNADIFFGATEPWL` and `LEF58_ANTENNAMAXCUMAREA`. This command supports checking antenna rules on the backside metal areas for both PG and signal nets.

Note: Before running this command, ensure that the process antenna or maximum floating area keywords are specified in the LEF file and the signal nets are routed.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `check_antenna` parameter. For a detailed description of the command and all its parameters, use the `man` command: `man check_antenna`
`-detailed`	Generates a report containing the PAE value for all layers of all the nets in a design, including those with no violations. If this parameter is not set, the report will include only those layers in the net that have violations. **Note:** To know the point of violation in the report, use the * mark. *Data_type*: bool, optional
`-error integer`	Specifies the maximum number of error messages to report. *Default:* `1000` *Data_type*: int, optional
`-nets {net_names}`	Reports the specified named nets. If you specify one net name, you do not need the braces around the name. *Data_type*: string, optional
`-out_file file_name`	Specifies the name of the output file. The file contains the PAE violation information. *Default:* *design_name*`.antenna.rpt` *Data_type*: string, optional
`-selected`	Reports the specified selected nets. Select nets using the `select_obj` command. If you specify one net name, you do not need the braces around the name. *Data_type*: bool, optional
`-use {all | signal | power | clock}`	Specifies the type of net to be checked. *Data_type*: enum, optional

Example

The following command verifies PAE violations in the routed design and generates a report file called `Test7.antenna.rpt`:

```
check_antenna -report Test7.antenna.rpt
```

Related Information

- `check_process_antenna`
- The following sections in the Identifying and Viewing Violations chapter in the *Innovus User Guide*
  - Verifying Process Antennas
  - Verifying Well-Process-Antenna Violations

  - Verifying Maximum Floating Area Violations
- The Repairing Process Antenna Violations section in the Using the NanoRoute Router chapter in the *Innovus User Guide*
- The following sections in the *LEF/DEF Language Reference* document:
  - Calculating and Fixing Process Antenna Violations
  - LEF Syntax

check_connectivity

```
check_connectivity
[-help]
[-add_new_open_markers]
[-append]
[-bbox_open_markers]
[-check_pg_ports]
[-check_wire_loops | -check_geometry_loops | -geometry_connect]
[-delete_old_open_markers]
[-die_abstract_file abstractdiefilenames]
[-divide_power_nets]
[-error integer]
[-ignore_dangling_wires]
[-ignore_floating_metal]
[-ignore_opens]
[-ignore_pg_pin cell_name]
[-ignore_soft_pg_connects]
[-ignore_unconnected_pins]
[-ignore_unrouted_nets]
[-ignore_weak_connects]
[-marker_on_highest_layer]
[-nets netNames | -selected]
[-no_fill]
[-out_file filename]
[-preferred_top_layer integer]
[-type {all | special | regular}]
[-use_external_connects]
[-warning value]
```

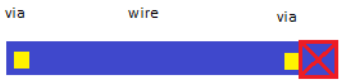
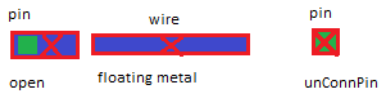
Detects conditions such as opens, unconnected wires (geometric antennas), unconnected pins, loops, partial routing, and unrouted nets; generates violation markers in the design window; reports violations. Running this command does not have database impact unless you save the design, which also saves the violation markers.

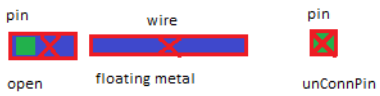
The Verify Connectivity feature now uses `set_multi_cpu_usage` and other multi-CPU commands for multi-threading.

For more information, see the Multiple-CPU Processing Commands chapter in the *Innovus Stylus Common UI Text Command Reference* document.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `check_connectivity` parameter. For a detailed description of the command and all its parameters, use the `man` command: `man check_connectivity`
`-add_new_open_markers`	Uses an open violation marker to show unconnected objects. *Data_type*: bool, optional
`-append`	Displays incremental results in the Violation Browser. By default, violation markers are over-written with new results during each `check_connectivity` run. When you specify this parameter, violation markers are appended to the violations of the previous `check_connectivity` run. *Data_type*: bool, optional
`-bbox_open_markers`	Displays violation markers for opens as the bounding box of the island. By default, violation markers for opens are displayed as polygons that include all wires, pins, and vias that connect to the island. *Data_type*: bool, optional
`-check_geometry_loops`	Checks for loop violations of regular nets using a geometrical model. The nets do not have to overlap on the center line. When you specify this parameter, Innovus does not perform any other connectivity checks. Use this parameter if you use a third-party router that does not route using the centerline connection routing technique. In this case, the `-check_wire_loops` parameter might not detect connectivity loop violations. *Data_type*: bool, optional

`-check_pg_ports`	Verifies all PG ports connection. *Data_type*: bool, optional
`-check_wire_loops`	Checks for connectivity loops in regular wires. Detects connectivity loops based on the end points of the center line of a regular wire segment or the center of a via. *Data_type*: bool, optional
`-delete_old_open_markers`	Removes old open violations. *Data_type*: bool, optional
`-die_abstract_file` *`abstractdiefilenames`*	Specifies the TSV abstract die files to be used to check the uBumps between the current die and the adjacent dies. Here, *`abstractdiefilenames`* specifies the names of the abstract die files created using the command `write_die_abstract`. *Data_type*: string, optional
`-divide_power_nets`	Divides power nets into four subareas for connectivity verification. Use this parameter to decrease memory usage in 32-bit machines with limited memory. This parameter might increase or decrease the run time, depending on the design. *Data_type*: bool, optional
`-error` *`integer`*	Specifies the maximum number of errors to report. The command stops when the maximum value is reached. *Default:* `1000` *Data_type*: int, optional
`-geometry_connect`	Checks for connectivity violations of regular wires. Uses a geometrical model instead of the center-line model. In other words, if the wires overlap at any point, they are considered to be connected--they do not have to connect at the center line. Use this parameter if you manually change the routing or use a third-party router that does not route using the centerline connection routing technique. *Data_type*: bool, optional
`-ignore_dangling_wires`	Ignores violations due to unconnected wires (also called geometrical antennas or dangling wires). A wire is called an antenna when the wire end to via center distance is less than ½ of the wire width as shown in the picture below:  In this picture, the blue rectangle is the wire and the yellow squares are the vias. *Data_type*: bool, optional
`-ignore_floating_metal`	Ignores floating metals. A wire is called floating methal, when both ends of the wire are not connected to pin shapes as shown in the picture below. In this picture, the blue rectangle is the wire and the green squares are pin shapes. *Data_type*: bool, optional
`-ignore_opens`	Ignores open violations. A wire is said to be open when one of its ends is connected to a pin shape but the other end is not connected to any pin as shown in the picture below.  In this picture, the blue rectangle is the wire and the green squares are pin shapes. **Note:** This parameter automatically turns on the `-ignore_unconnected_pins` parameter, because open and unconnected pin verification are performed together. A pin shape that is not connected to any wire is called an unconnected pin (`unConnPin`). *Data_type*: bool, optional

`-ignore_pg_pin` *`cell_name`*	Specifies the cell containing the PG pin that can be ignored during connectivity check. *Data_type*: string, optional
`-ignore_soft_pg_connects`	Disables checking of soft Power/Ground connects. By default, `check_connectivity` checks the connectivity on masterslice layers. If `-ignore_soft_pg_connects` option is specified, connectivity on these layers is ignored. *Data_type*: bool, optional
`-ignore_unconnected_pins`	Ignores violations due to pins that are not connected to any other objects. In the picture below, the pin on the right is an unconnected pin (`unConnPin`) as it is not connected to any wire.  **Note:** This parameter is specified automatically when you specify the `-ignore_opens` parameter. *Data_type*: bool, optional
`-ignore_unrouted_nets`	Ignores nets that are not routed. *Data_type*: bool, optional
`-ignore_weak_connects`	Disables checking for routing to more than one port of the weakly connected pin ports. *Data_type*: bool, optional
`-marker_on_highest_layer`	Reports the opening on the highest layer of the pin, which has more one shape. *Data_type*: bool, optional
`-nets` *`netNames`*	Specifies that connectivity should be verified for specified nets only. If you specify more than one net, separate the net names with space and surround the list of net names with braces or quotes. You can use wildcards (`*` and `?`) when you specify net names. **Note:** The `-nets` and `-selected` parameters are mutually exclusive. *Default:* If you do not specify either the `-nets` or the `-selected` parameter, `check_connectivity` verifies all nets. *Data_type*: string, optional
`-no_fill`	Specifies that metal fill also needs to be checked. *Data_type*: bool, optional
`-out_file` *`filename`*	Specifies the report file for connectivity violation data. *Data_type*: string, optional
`-preferred_top_layer` *`integer`*	Specifies the preferred top metal layer. When this parameter is used, check_connectivity does not check the specified top layer and above layers. *Data_type*: int, optional
`-selected`	Specifies that connectivity should be checked for selected nets or the nets connected to the pins of selected instances/macros. **Note:** The `-nets` and `-selected` parameters are mutually exclusive. *Default:* If you do not specify either the `-nets` or the `-selected` parameter, `check_connectivity` verifies all nets in the design. *Data_type*: bool, optional

`-type {all | regular | special}`	Specifies the type of wires to verify. `check_connectivity` is only concerned with the type of the wire and via (special or regular) and not the type of the pin. Irrespective of the `type` specified, all unconnected terminals (special and regular) are always reported unless `-ignore_unconnected_pins` is on. *Default:* `all` **Note:** To check connectivity for whole nets, you do not need to specify a type. **Note:** A pin-open violation is listed under VC DRC type "Problem(s) (ENCVFC-92): Pieces of the net are not connected together". Such a violation means the isolated open island only includes some pins' shapes that come from at least two different ports or pins. Choose one of the following: `all`: Checks all wires, including those that have been previously verified. `regular`: Reports all unconnected pins and open islands that do not include any special wire and via. Does not check a net that has no regular objects (wire, via, pin). `special`: Reports all unconnected pins and open islands that do not include any regular wire and via. Does not report isolated open islands that have any regular wire and via. *Data_type*: enum, optional
`-use_external_connects`	Implies a virtual connection for all bumps and external I/O pins of the same net. Set this parameter to override default behavior of `check_connectivity`, in which bumps and external I/O pins of the same net are not considered to be interconnected. It is useful in flip chip designs, where the power bumps are connected outside the chip. *Data_type*: bool, optional
`-warning` *value*	Specifies the maximum number of warnings to report. *Default:* `50` *Range:* `0` to `1000000` *Data_type*: int, optional

Examples

- The following command verifies connectivity for all nets and generates a maximum of 50 error and 50 warning messages:
  ```
  check_connectivity -type all -error 50 -warning 50
  ```
- The following command reports special unconnected pins and open islands that do not include any regular wire and via
  ```
  check_connectivity -type special -ignore_dangling_wires -error 1000 -warning 50
  ```

Related Information

The following sections in the Identifying and Viewing Violations chapter in the *Innovus User Guide*:

- Verifying Connectivity
- Viewing Violations With the Violation Browser

check_cut_density

```
check_cut_density
[-area {x1 y1 x2 y2}]
[-detailed]
[-global_density]
[-ignore_cell_obs]
[-layers layer_name_list]
[-oversize value]
[-out_file fileName]
```

Checks the density of specified cut layers or areas of cut layers, or the cut density of the whole chip.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `check_cut_density` parameter. For a detailed description of the command and all its parameters, use the `man` command: `man check_cut_density`
`-area {x1 y1 x2 y2}`	Specifies the coordinates of the area to verify. For example, `check_cut_density -area {0 0 100 100}`. *Default:* Entire area *Data_type*: rect, optional
`-detailed`	Generates a detailed cut density report. *Data_type*: bool, optional
`-global_density`	Checks the density across the entire design for the specified cut layers and writes the results to the report file. *Data_type*: bool, optional
`-ignore_cell_obs`	Disables checking of windows that are fully covered by block or pad obstructions. *Data_type*: bool, optional
`-layers layer_name_list`	Specifies the cut layers to verify. Use the following format to specify layer names: {Via12 Via23} For example, `check_cut_density -layers {Via12 Via23}`. *Data_type*: string, optional
`-out_file fileName`	Specifies the report file for the cut density violation information. *Default:* *designName*`.cutdensity.rpt` *Data_type*: string, optional
`-oversize value`	Specifies an offset value for the area to verify. The value is in user units (not in DBU), and can be positive or negative. A positive value adds to the area that is verified. For example, if the design covers the area from `(0,0)` to `(100,100)` and you specify `-oversize 10`, `check_cut_density` checks the area from `(-10,-10)` to `(110,110)`. *Default:* `0` *Data_type*: int, optional

Example

- The following command checks the density across the entire design for the layers `Via12` and `Via23` and generates a detailed report named `density.rpt`.
  ```
  check_cut_density -layers {Via12 Via23} -out_file density.rpt -detailed -global_density
  ```

Related Information

- Identifying and Viewing Violations chapter in the *Innovus User Guide*

  - Overview
- Optimizing Metal Density chapter in the *Innovus User Guide*
  - Overview
  - Adding Via Fill
- LEF Syntax chapter in the *LEF/DEF Language Reference*

check_drc

```
check_drc
[-help]
[-area {{lx1 ly1 ux1 uy1} {lx2 ly2 ux2 uy2}...}]
[-check_cuts_in_same_via]
[-check_drc_check_reverse]
[-check_ndr_spacing]
[-check_only {all | regular | special | selected_net | selected | cell  | default}]
[-check_routing_halo]
[-check_short_only]
[-check_trim_length]
[-check_uncolored]
[-enable_post_passive_fill_check]
[-exclude_pg_nets]
[-ignore_trial_route]
[-layer_range {layer1 [layer2]}]
[-limit max_error]
[-out_file file_name]
[-view_window]
```

Checks for DRC violations and creates violation markers in the design database that can be seen on the GUI and browsed with the Violation Browser.

This command uses a new DRC engine that can support all the tech nodes. You can use the `set_db` command to set global parameters for `check_drc`. The parameters you specify with the `set_db` command are then used as default automatically whenever you run `check_drc`. The settings are saved in the design so you need not set them each time you run `check_drc`.

Use `get_db` or `report_obj -categories check_drc -all` to return the current settings for the `check_drc` command. Use `get_db -category check_drc` to list available attributes to set.

Note: The `check_drc` command checks only the placed instances. It skips checking the DRC for unplaced instances.

Note: The `check_drc` command adds the LEF rule in the description section of the *Violation Browser* to provide more information on the rule that is causing the DRC violation.

Parameters

`-help`	Prints a brief description that includes the type and default information for each `check_drc` parameter. For a detailed description of the command and all its parameters, use the man command: `man check_drc`
`-area {{lx1 ly1 ux1 uy1} {lx2 ly2 ux2 uy2}...}`	Specifies the coordinates of one or more area regions to verify. For example, `check_drc -area {0 0 100 100} {100 100 200 200}`. *Data_type*: rect, optional
`-check_cuts_in_same_via`	Checks for any cut-to-cut spacing violation between cuts from the same via cell. *Data_type*: bool, optional
`-check_drc_check_reverse`	Reports the `CONVEXCORNER REVERSE` violation in the flow. *Data_type*: bool, optional
`-check_ndr_spacing`	Checks non-default rule spacing as a hard rule. *Data_type*: enum, optional

`-check_only {all | regular | special | selected_net | selected | cell | default}`	Specifies which types of shapes to be checked: • `all`: Reports violations between all routing shapes (regular and special) and all other shapes (including routing blockages, and cell pin/obs shapes). • `regular`: Only reports violations between regular routing shapes (normally created by nanoroute) and all other shapes. It does not report violations between special routes, routing blockages and cell shapes. This is useful to find violations caused only by nanoroute. • `special`: Only reports violations between special routes, and all other shapes. It does not report violations between regular routes, routing blockages, and cell shapes. This is useful when checking power routing or pre-routed special routes, and isolating these errors from other violations. • `selected_net`: Only reports violations between the selected list of nets. Use this parameter value when the DRC fixes are done on nets and complete/area based DRC check is time consuming or it flags multiple other violations. • `selected`: Reports only the violations on special wire objects/vias of the secondary PG nets with the specified user class. • `cell`: Only reports the cell. • `default`: This value is equivalent to specifying the `regular` and `special` at the same time. *Data_type*: enum, optional
`-check_routing_halo`	Checks for routing halo violations. Routing halos are used to avoid cross-coupling between top-level and inside-the-block routes. As short jogs do not affect cross-coupling much, they can be ignored even if they overlap with the halo region. Normally, it is better to have some short jogs instead of more vias. When the `-check_routing_halo` option is specified, any jog shape with length less than the value set for `set_db -check_drc_max_wrong_way_halo` is ignored for the wrong way check. `check_drc` reports violations only for the jog shapes that have lengths greater than the `set_db -check_drc_max_wrong_way_halo` setting. *Default:* If the `set_db -check_drc_max_wrong_way_halo` option has not been set, `check_drc` uses `4* pitch` as the default value for the routing halo, where `pitch` is the METAL2 routing pitch. *Data_type*: bool, optional
`-check_short_only`	Excludes checking all other DRCs except the short ones. *Data_type*: bool, optional
`-check_trim_length`	Checks the maximum length of the `LEF58_TRIMSHAPE` value for detecting the DRC violations in the trim shape layer. *Data_type*: bool, optional
`-check_uncolored`	Reports violations with no color and invalid mask color for the multi-pattern (MPT) layers. *Data_type*: bool, optional
`-enable_post_passive_fill_check`	Checks if the track is fully filled for the `full_fill` trim layer. *Data_type*: bool, optional
`-exclude_pg_nets`	Excludes the checking of power and ground (P/G) nets. `check_drc -exclude_pg_nets` reports violations caused by routing (special and regular) except for a P/G net. It also reports violations for implant layers. *Data_type*: bool, optional

`-ignore_trial_route`	Ignores trial route when checking drc. *Data_type*: bool, optional
`-layer_range {layer1 [layer2]}`	Checks between the specified range of metal layers, including the cut layers in between. You can use the full layer name, or abbreviations like `metal1` or `1` for the first routing layer, etc. For example, `-layer_range {metal1 metal3}` will check all the layers between `metal1` to `metal3`. If `layer2` is not specified, it only checks DRC on `layer1`. *Data_type*: string, optional
`-limit` *max_error*	Specifies the maximum number of errors to report. *Default:* 1000 *Data_type*: int, optional
`-out_file` *file_name*	Specifies the report file that contains the violations information. *Data_type*: string, optional
`-view_window`	Checks for DRC violations in the current window. *Data_type*: bool, optional

Examples

- Any one of the following commands checks for DRC violations between Metal2 and Metal3 layers:

```
check_drc -layer_range {metal2 metal3}
check_drc -layer_range {metal3 metal2}
```

- The following command checks for special wire related DRC violations, such as spacing violation between power/ground wire to cell blockage:

```
check_drc -check_only special
```

Related Information

- Identifying and Viewing Violations chapter in the *Innovus User Guide*
- `check_place`
- `delete_drc_markers`
- `write_drc_markers`

check_drc Category Attributes

The root attributes in the `check_drc` category are:

- check_drc_area
- check_drc_check_only
- check_drc_check_reverse
- check_drc_check_routing_halo
- check_drc_check_routing_halo_corner
- check_drc_disable_rules
- check_drc_enable_post_passive_fill
- check_drc_exclude_pg_net
- check_drc_ignore_cell_blockage
- check_drc_ignore_non_rectangle_shapes
- check_drc_ignore_trial_route
- check_drc_inside_via_def
- check_drc_layer_range
- check_drc_limit
- check_drc_max_wrong_way_halo
- check_drc_ndr_spacing
- check_drc_report
- check_drc_short_only
- check_drc_trim_length
- check_drc_uncolored
- check_drc_use_min_spacing_on_block_obs

To set the value of any root attribute, use the `set_db` command:

```
set_db attribute_name value
```

To get the current value of any root attribute, use the `get_db` command:

```
get_db attribute_name
```

To get the current value of all attributes in the `check_drc` category, use the following command:

```
get_db -category check_drc
```

check_drc_area

`check_drc_area {coordinates}`

Default: `{0.0 0.0 0.0 0.0}`

Data_type: rect, read/write

Specifies the coordinates of the area to verify.

Related Commands

`check_drc`

check_drc_check_only

`check_drc_check_only {all | regular | special | selected_net | selected | cell | default}`

Default: `all`

Data_type: enum {all regular special selected_net cell default}, read/write

Specifies which types of shapes are to be checked.

- `all`: This is the default value, and reports violations between all routing shapes (regular and special) and all other shapes (including routing blockages, and cell pin/obs shapes). This release has a limitation that it will not check for violations between cell pin/obs shapes (either inside one cell, or between two cells). You can check for cell overlap with `check_place` which normally finds any cell to cell violations.
- `regular`: Only reports violations between regular routing shapes (normally created by NanoRoute) and all other shapes. It does not report violations between special routes, routing blockages and cell shapes. This is useful to find violations caused only by NanoRoute.
- `special`: Only reports violations between special routes, and all other shapes. It does not report violations between regular routes, routing blockages, and cell shapes. This is useful when checking power routing or pre-routed special routes, and isolating these errors from other violations.
- `selected_net`: Reports only the violations between the selected list of nets. Use this attribute value when the DRC fixes are done on nets and complete/area based DRC check is time consuming or it flags multiple other violations.
- `selected`: Reports only the violations on special wire objects/vias of the secondary PG nets with the specified user class.
- `cell`: Reports only the cell.
- `default`: Reports violations between the default shapes.

Related Commands

`check_drc`

check_drc_check_reverse

`check_drc_check_reverse {true | false}`

Default: `false`

Data_type: bool, read/write

Reports the `CONVEXCORNER REVERSE` violation in the flow.

Related Commands

`check_drc`

check_drc_check_routing_halo

`check_drc_check_routing_halo {true | false}`

Default: `false`

Data_type: bool, read/write

Checks for routing halo violations.

Related Commands

`check_drc`

check_drc_check_routing_halo_corner

`check_drc_check_routing_halo_corner {true | false}`

Default: `false`

Data_type: bool, read/write

Checks for routing halo violations in the corner.

Related Commands

`check_drc`

check_drc_disable_rules

`check_drc_disable_rules` *`rule_name`*

Default: ""

Data_type: enum* {jog2jog_spacing eol_spacing cut_spacing min_cut enclosure color min_step protrusion min_area out_of_die off_manufacturing_grid off_routing_track}, read/write

Disables the specified rule.

Related Commands

`check_drc`

check_drc_enable_post_passive_fill

`check_drc_enable_post_passive_fill {true | false}`

Default: `false`

Data_type: bool, read/write

Checks if the track is fully filled for the `full_fill` trim layer.

Related Commands

`check_drc`

check_drc_exclude_pg_net

`check_drc_exclude_pg_net {true | false}`

Default: `false`

Data_type: bool, read/write

Excludes the checking of power and ground nets.

Related Commands

`check_drc`

check_drc_ignore_cell_blockage

`check_drc_ignore_cell_blockage {true | false}`

Default: `false`

Data_type: bool, read/write

Ignores geometries of cell blockage.

Related Commands

`check_drc`

check_drc_ignore_non_rectangle_shapes

`check_drc_ignore_non_rectangle_shapes {true | false}`

Default: `true`

Data_type: bool, read/write

When set to `true`, this attribute ignores checking DRC markers for the objects with the non-rectangular shapes. When set to `false`, it checks DRC markers for the objects of all shapes.

Related Commands

`check_drc`

check_drc_ignore_trial_route

`check_drc_ignore_trial_route {true | false}`

Default: `false`

Data_type: bool, read/write

Ignores trial route when checking drc.

Related Commands

`check_drc`

check_drc_inside_via_def

`check_drc_inside_via_def {true | false}`

Default: `false`

Data_type: bool, read/write

Checks the cut spacing from the same via.

Related Commands

`check_drc`

check_drc_layer_range

`check_drc_layer_range {layer1 [layer2]}`

Default: `""`

Data_type: string, read/write

Checks between the specified range of metal layers, including the cut layers in between. You can use the full layer name, or abbreviations like metal`1` or `1` for the first routing layer, etc. If *`layer2`* is not specified, it only checks DRC on *`layer1`*. For example, the following command checks all the layers between `metal1` and `metal3`:

```
set_db check_drc_layer_range {metal1 metal3}
```

Related Commands

`check_drc`

check_drc_limit

`check_drc_limit value`

Default: `1000`

Data_type: int, read/write

Specifies the maximum number of errors to report.

Related Commands

`check_drc`

check_drc_max_wrong_way_halo

`check_drc_max_wrong_way_halo` *`halo_length`*

Default: `""`

Data_type: double, read/write

Specifies the length up to which short jog shapes in halo area can be ignored. Routing halos are used to avoid cross-coupling between top-level and inside-the-block routes, and short jogs do not affect cross-coupling much. Normally, it is better to have some short jogs rather than more vias. When this option is specified, any segment with length less than *`halo_length`* is ignored for wrong way check, even if it overlaps with the halo region. `verify_drc` reports violations only for jog shapes with lengths greater than *`halo_engths`*.

Default: If you do not set the `check_drc_max_wrong_way_halo` attribute, `check_drc` uses 4* *`pitch`* as the default value, where *`pitch`* is the METAL2 routing pitch.

Related Commands

`check_drc`

check_drc_ndr_spacing

`check_drc_ndr_spacing {true | false | auto}`

Default: `auto`

Data_type: enum {true false auto}, read/write

Checks non-default rule spacing as a hard rule. Following are the attribute values:

- `true`: Checks all non-default rules.
- `false`: Skips checking any non-default rule.
- `auto`: Checks only the non-default rules with `hard_spacing`.

Related Commands

`check_drc`

check_drc_report

`check_drc_report` *`filename`*

Default: `""`

Data_type: string, read/write

Specifies the report file that contains the violation information.

Related Commands

`check_drc`

check_drc_short_only

`check_drc_short_only {true | false}`

Default: `false`

Data_type: bool, read/write

Excludes checking all other DRCs except the short ones.

Related Commands

`check_drc`

check_drc_trim_length

```
check_drc_trim_length {true | false}
```

Default: `false`

Data_type: bool, read/write

Checks the maximum length of the `LEF58_TRIMSHAPE` value for detecting the DRC violations in the trim shape layer.

Related Commands

`check_drc`

check_drc_uncolored

```
check_drc_uncolored {true | false}
```

Default: `false`

Data_type: bool, read/write

Reports color change violation when the DPT layer shape is 'free color'.

Related Commands

`check_drc`

check_drc_use_min_spacing_on_block_obs

```
check_drc_use_min_spacing_on_block_obs {0 | 1 | auto}
```

Default: `auto`

Data_type: enum, read/write

Checks all the Macro OBS with minSpacing.

Related Commands

`check_drc`

check_endcaps

```
check_endcaps
[-help]
[-area {x1 y1 x2 y2}]
[-core_boundary_only]
[-error integer]
[-ignore_macro SCLA,...]
[-out_file filename]
[-power_domains string]
[-rows string]
[-triple_well]
[-wrong_location]
```

Checks whether pre/post cap cells have been inserted correctly based on `set_db` settings. Use this command after running `add_endcaps` to flag violations such as:

- Inserted end cap cell is of wrong type
- End cap cell is missing

If you use `check_endcaps` with the `-wrong_location` option, it also checks whether the end cap cells are inserted in the right location. Use the `-power_domains` option to check end cap cells inserted for a specific power domain.

Additionally, you can use `check_endcaps -triple_well` to check triple well insertion. Triple well technology is used to isolate p-well from substrate using a deep n-well.

You can view the violation markers generated by `check_endcaps` in Violation Browser.

Parameters

`-help`	Prints a brief description that includes the type and default information for each `check_endcaps` parameter. For a detailed description of the command and all its parameters, use the `man` command: `man check_endcaps`
`-area {x1 y1 x2 y2}`	Specifies the area in which endcaps are to be verified. *Data_type*: rect, optional
`-core_boundary_only`	Checks endcaps only at the core-boundary and not abutting obstructions. *Data_type*: bool, optional
`-error integer`	Specifies the maximum number of errors to report. *Data_type*: uint, optional
`-out_file filename`	Specifies the file name for the `check_endcaps` report. *Data_type*: string, optional
`-ignore_macro SCLA,...`	Specifies the macro to be ignored. *Data_type*: string, optional
`-power_domains string`	Specifies the power domain to be checked. Use this option to check end cap cells inserted for the specified power domain. *Data_type*: string, optional
`-rows string`	Specifies the row to check. *Data_type*: string, optional
`-triple_well`	Checks triple well insertion. Only checks the cell lists specified in `set_db`. *Data_type*: bool, optional
`-wrong_location`	Checks for cap cells that are not inserted in the right location. If you specify this option, `check_endcaps` marks with a violation marker any cap cell that is not on the beginning/end of the row or not on the boundary of the design/ block design. *Data_type*: bool, optional

Examples

- The following set of commands verifies whether cap cell insertion is as per the `set_db` settings and generates a default report file in the active folder:

```
set_db -add_endcaps_left_edge EC_LE10 -add_endcaps_right_edge EC_RE10
check_endcaps
```

- The following command verifies end cap insertion and generates a report named `endcap.rpt`:

```
set_db -add_endcaps_left_edge EC_LE10 -add_endcaps_right_edge EC_RE10
check_endcaps -out_file endcap.rpt
```

- The following command verifies triple well insertion. It also checks the cap cells that are not inserted in the right location and reports up to `10000` errors:

```
set_db -add_endcaps_left_edge EC_LE10 -add_endcaps_right_edge EC_RE10 -add_endcaps_bottom_edge EC_BE10 \
              -add_endcaps_left_bottom_corner EC_LBC10 -add_endcaps_left_top_corner EC_LTC10 \
              -add_endcaps_right_bottom_corner EC_RBC10 -add_endcaps_right_top_corner EC_RTC10 \
              -topEdge EC_TE10
check_endcaps -triple_well -wrong_location -error 10000
```

Related Information

- "Verifying End Cap Violations" section in the Identifying and Viewing Violations chapter of the *Innovus User Guide*
- "Adding End-Cap Cells" section in the Placing the Design chapter of the *Innovus User Guide*

check_io_to_bump_connectivity

```
check_io_to_bump_connectivity
[-help]
[-error integer]
[-report string]
[-sub_class string]
[-warning integer]
[-bump_connect_target | -class_bump ]
[-bumps string | -class_bump ]
[-bumps string | -nets string]
```

Verifies multiBump multiPad connectivity. The flip chip flow in Innovus supports multiple pads to multiple bumps connections. Use the `check_io_to_bump_connectivity` command, instead of `check_connectivity` to verify such connections.

Parameters

`-help`	Prints a brief description that includes the type and default information for each `check_io_to_bump_connectivity` parameter. For a detailed description of the command and all its parameters, use the `man` command: man `check_io_to_bump_connectivity`
`-bump_connect_target`	Checks only whether or not the routing matches the connection specified by `bump_connect_target` properties (`io:pin:port`) regardless of the class of the port. *Data_type*: bool, optional
`-bumps string`	Specifies the name of the bumps that are to be checked. *Data_type*: string, optional
`-class_bump`	Checks only whether or not the ports of `CLASS BUMP` are routed to the bumps regardless of the `bump_connect_target` properties. *Data_type*: bool, optional
`-error integer`	Specifies the maximum number of errors to report. *Data_type*: int, optional
`-nets string`	Specifies the names of the nets that are to be checked. *Data_type*: string, optional
`-report string`	Specifies the name of the report file in which `check_io_to_bump_connectivity` violation data is recorded. *Data_type*: string, optional
`-sub_class string`	Specifies the name of the subclasses of which the wires and vias are to be checked. *Data_type*: string, optional
`-warning integer`	Specifies the maximum number of warnings to report. *Data_type*: int, optional

Related Information

The Identifying and Viewing Violations chapter in the *Innovus User Guide*.

check_isolated_cuts

```
check_isolated_cuts
[-help]
[-area {x1 y1 x2 y2}]
[-error integer]
[-layer {cutLayer_list}]
[-spacing_bbox]
[-report filename]
```

Checks cuts in all or specified cut layers against the spacing rules set by `set_isolated_cut_rule`. If you use the `-area` parameter, checks cuts within the specified area. Reports violations in the log file and in `verify.isoVia.rpt` or the specified report file.

Note: Violations can be viewed in the Violation Browser.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `check_isolated_cuts` parameter. For a detailed description of the command and all its parameters, use the `man` command: `man check_isolated_cuts`
`-area {x1 y1 x2 y2}`	Specifies the coordinates of the area to verify. If you do not specify this parameter, the command checks the whole design. For example, `check_isolated_cuts -area {0 0 100 100}`. *Default:* Entire area *Data_type*: rect, optional
`-error integer`	Specifies the maximum number of errors to report. *Default:* `1000` *Data_type*: int, optional
`-layer {cutLayer_List}`	Specifies the cut layers to be checked. If you specify only one layer name, you do not need the braces around the name. For example, `check_isolated_cuts -layer {Via1 Via2}`. *Default:* All cut layers *Data_type*: string, optional
`-report filename`	Specifies the report file for the isolated cut violation information. *Default:* `verify.isoVia.rpt` *Data_type*: string, optional
`-spacing_bbox`	Specifies that the Square metric should be used for checking isolated cuts. By this metric, another cut should exist within a square window from a certain cut. The size of the square window depends on the spacing rules set by `setIsolatedCutRule`. **Note:** If this parameter is not specified, the Euclidean metric is used for checking isolated cuts. *Data_type*: bool, optional

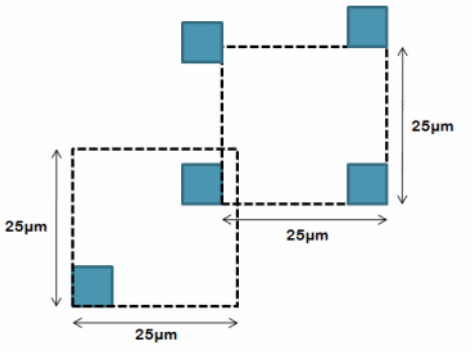
Example

- The following command checks isolated cuts on all layers and generates a report file named `test.isoCut.rpt`:
  ```
  verifyIsolatedCut -report test.isoCut.rpt
  ```
- The following command checks for isolated cuts in the area specified and generates a report named `verify_isoVia.rpt`:
  ```
  check_isolated_cuts -area {0 0 1600 1600}
  ```
- The following set of commands check for isolated cuts on all layers using the Square metric. Any cut that does not have another cut within a 25um x 25um square window is reported:

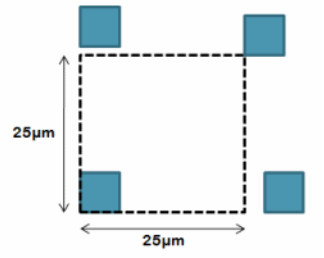
  [DEV]innovus 45> setIsolatedCutRule -spacing 25

  [DEV]innovus 46> check_isolated_cuts -spacing_bbox

  In the following pattern, no isolated cut violation is reported as each cut has one or more cuts within a `25um x 25um` square window.



In contrast, the following pattern is not okay because no other cut exists within a 25um x 25um square window:



Related Information

Verifying Isolated Cuts section in the Identifying and Viewing Violations chapter of the *Innovus User Guide*

check_litho

```
check_litho
[-help]
-apply {hotspot | ripandreroute | squish_hints}
[-check_function string]
[-check_local]
[-config string]
[-cpu integer]
[-create_guides]
-dir string
[-hints_to_apply string]
[-incremental_check string]
[-layer_map_file string]
[-map_file string]
[-no_compress]
[-oasis]
[-oasis_files string]
[-option_file string]
[-prev_run_dir string]
[-report_file string]
{-read_blockage | -report_only | -routing_layers_only | -sign_off}
[-run_mode string]
[-stream_files string]
[-write_stream_options string]
[-techfile string]
```

Runs LPA on the selected design to flag litho hotspots.

Note: The `check_litho` command is available only if you have the appropriate license and the MVS installation path set up. Refer to the Prerequisites for Running LPA chapter of *Innovus User Guide*.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `check_litho` parameter. For a detailed description of the command and all its parameters, use the `man` command: `man check_litho`
`-apply {hotspot | ripandreroute | squish_hints}`	
	Instructs Innovus to perform specific operations based on the results of `check_litho`. The operation performed depends on the value you specify with `-apply`: • `hotspot`: Loads the resulting hotspots into violation browser and prepare for regular litho fixing. • `ripandreroute`: Sets up Innovus to fix hotspots using the router with sized and shifted locations from the hotspot marker. • `squish_hints`: Applies the fill and via replace hints created by `check_litho` automatically if `check_litho` is being run using squish patterns.
`-check_function string`	Specifies the DRC function to be checked.
`-check_local`	Applies fixing with the local DRC.
`-config string`	Specifies the name and path of the LPA custom configuration file. This file, if specified, controls all run options of LPA. It can also specify different instructions for each layer.
`-cpu integer`	Specifies the total number of CPUs available for the LPA run.
`-create_guides`	Generates LPA guidelines.
`-dir string`	Specifies the LPA output directory. This directory contains one subdirectory for each layer. The individual subdirectories for each layer (POLY, 1, etc.) are created when LPA is run with the configuration file for that layer.
`-hints_to_apply string`	Specifies the list of hint number to apply.

`-incremental_check` *string*	Specifies the name and path of the HIF file that you want to use for incremental validation. This HIF file includes the locations that identify the areas affected by each hotspot fix. LPA reads these locations and performs incremental checking only in these areas. This reduces the time for validation.
`-layer_map_file` *string*	Specifies the name of the layer map file.
`-map_file` *string*	Specifies the name and path of the Stream Out Map file created by Innovus to map the GDS stream to the layers in the Innovus database. If the GDS List file and Stream Out Map file are specified, LPA runs on the potential IP blocks defined in these files.
`-no_compress`	Inputs `gds` instead of `gds.gz`.
`-oasis`	Streams out using oasis.
`-oasis_files` *string*	Specifies the `oasis_out` oasis list files.
`-option_file` *string*	Specifies a text file, which can be used to set variables used by the `-apply` operations.
`-prev_run_dir` *string*	Specifies the path to the run directory of a previous Verify Litho run that you want to use for XOR-based incremental validation. When this parameter is specified, the previous run results are compared to the new design and LPA is run only in locations where the layout has changed and where hotspots previously existed, thereby reducing the overall validation run time. This flow is recommended when the design has been changed but no HIF file that identifies the changed areas is available.
`-read_blockage`	Loads the blockage file. **Note**: This parameter cannot be used along with `-sign_off`, `-routing_layers_only`, or `-read_blockage`.
`-report_file` *string*	Specifies the netlist report file.
`-report_only`	Generates the netlist report. **Note**: This parameter cannot be used along with `-sign_off`, `-routing_layers_only`, or `-read_blockage`.
`-routing_layers_only`	Specifies that LPA is to be run in the Routing Layers Only mode. The Routing Layers Only mode is the fast mode of LPA to flag L1 hotspots in a design at the block level during the Implementation phase. In this mode, LPA runs about 100X faster than the Sign-Off mode and helps you identify and fix most hotspots as routing is completed. **Note**: This parameter cannot be used along with `-sign_off`, `-report_only`, or `-read_blockage`.
`-run_mode` *string*	Launches LPA in back/foreground.
`-sign_off`	Specifies that LPA is to be run in the Sign-Off mode. You run LPA in Sign-Off mode for Litho sign-off, as mandated by foundries. Unlike the Routing Layers Only mode, the Sign-Off mode requires you to provide the LPA configuration file containing the Techfile and other settings. You must run LPA Sign-Off mode before handing off the design to the top-level or tape-out. **Note**: This parameter cannot be used along with `-routing_layers_only`, `-report_only`, or `-read_blockage`.
`-stream_files` *string*	Specifies the name and path of the text file containing the list of GDS files for LEF abstracts. By default, LPA runs on the interconnect layers in Sign-Off mode but if the GDS List file and Stream Out Map file are specified, LPA runs on the potential IP blocks defined in these files.
`-techfile` *string*	Specifies the path and name of the qualified LPA Mx technology file that includes process-specific LPA model, hotspot patterns, and hotspot checking options. The LPA model captures the GDSII-to-silicon flow of the targeted process, enabling LPA to predict the variations and hotspots due to lithography, directly from the design database. Hotspot patterns can also be included in the LPA techfile in replacement for the LPA model to support the DRC+ flow, or along the LPA model to enable fast mode. **Note:** The `-techfile` parameter is mandatory if you use the `-routing_layers_only` option to run LPA.
`-write_stream_options` *string*	Enables additional `write_stream` command options.

Examples

Following is an example of the `check_litho` command usage:

```
check_litho -dir lpa -techfile lpa.tcl -layer_map_file lef_map.file -map_file gds_map.file -routing_layers_only -write_stream_options "-outputMacros -merge list_of_GDS_files"
```

Related Information

- Integration with LPA and CCP chapter of the *Innovus Stylus Common UI User Guide.*
- *Litho Physical Analyzer User Guide*

check_metal_density

```
check_metal_density
[-help]
[-report filename]
[-detailed]
[-ignore_lef_density]
[-layer layer_name_list]
[-area {x1 y1 x2 y2}]
[-oversize value]
```

Checks the metal density of each routing layer and of macros against values specified by `set_metal_fill`, by LEF file, or against its own internal default values. Reports density information in the log file.

> The `set_metal_fill` command is a user-override command that provides constraints at the run time for each session. These constraints have precedence over the LEF/database values. For example, if you specify `set_metal_fill -iteration_name default`, `check_metal_density` checks against the `default` value.

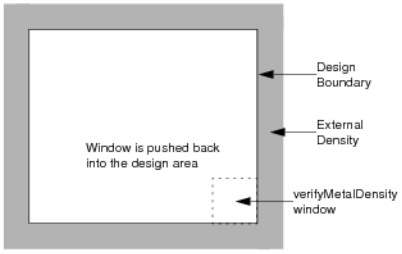
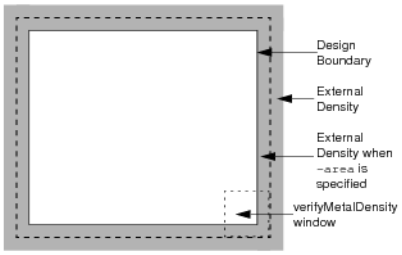
If neither the LEF file nor `set_metal_fill` specifies the window size or density, `check_metal_density` uses internal default values.

`check_metal_density` now supports multi-threading to improve Metal Fill runtime performance in multi-thread mode. `check_metal_density` supports the new `set_multi_cpu_usage` command use model. For more information, see the following documents:

- Multiple-CPU Processing Commands chapter of the *Innovus Stylus Common UI Text Command Reference*.
- Accelerating the Design Process by Using Multiple-CPU Processing chapter of the *Innovus User Guide*.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `check_metal_density` parameter. For a detailed description of the command and all its parameters, use the `man` command: `man check_metal_density`

`-area {x1 y1 x2 y2}`	Specifies the coordinates of the area to verify. For example, `check_metal_density -area {0 0 100 100}`. *Default:* Entire area **Behavior of `check_metal_density -area` with external metal density** When metal fill is added to a block design the external metal density provides a density assumption to `check_metal_density`. The external density is used only for density calculation purposes and no fill shapes are added outside of the design area. When the `-area` parameter is not specified, the external density value is not used. In this case, the `check_metal_density` window is pushed back if it extends outside the design area, as shown in the figure below.  If the `-area` parameter is specified and is larger than the design size then windows extend outside of the block area and use the external density value (see `set_metal_fill` external density value specification) for that area.  If external density is less than the specified preferred density, it is more likely that shapes will be added to the area. If external density is greater than the preferred density it is less likely that shapes will be added (shapes are only added within the design boundary). **Note:** Windows outside the design boundary are ignored. *Data_type*: rect, optional
`-detailed`	Generates a detailed metal density report. *Data_type*: bool, optional
`-ignore_lef_density`	Ignores the MACRO `DENSITY` table in the LEF file and instead uses the default macro density calculation method. **Note:** If you do not specify this parameter, `check_metal_density` uses the LEF MACRO density table for density calculations. *Data_type*: bool, optional
`-layer layer_name_list`	Specifies the LEF or Innovus layers to verify. If you specify only one layer name, you do not need the braces around the name. For example, `check_metal_density -layer {METAL1 METAL2}`. *Data_type*: string, optional
`-oversize value`	Specifies an offset value for the area to verify. The value is in user units (not in DBU), and can be positive or negative. A positive value adds to the area that is verified. For example, if the design covers the area from `(0,0)` to `(100,100)` and you specify `-oversize 10`, `check_metal_density` checks the area from `(-10,-10)` to `(110,110)`. *Default:* `0`. *Data_type*: int, optional
`-report filename`	Specifies the report file for the metal density violation information. *Default:* *designName*`.metaldensity.rpt`. *Data_type*: string, optional

Example

- The following command checks metal density of all layers and generates a report file named `Test7.density.rpt`:
  ```
  check_metal_density -report Test7.density.rpt
  ```
- The following command checks metal density on layer *Metal1* in the area specified and generates a detailed report named `test_chip.density.rpt`:
  ```
  check_metal_density -area {-200 -200 1600.0 1600.0} -layer {Metal1} -reportfile test_chip.density.rpt -detailed
  ```

  In the example above, the layout size is 0 0 1400.0 1400.0. The command considers the density of the area outside of the design as `0`.

Related Information

- Identifying and Viewing Violations chapter in the *Innovus User Guide*
- Optimizing Metal Density chapter in the *Innovus User Guide*
- LEF Syntax chapter in the *LEF/DEF Language Reference*

check_pg_shorts

```
check_pg_shorts
[-help]
[-area {x1 y1 x2 y2}]
[-net net_name]
[-no_cell_blockages]
[-no_route_blockages]
[-out_file filename]
```

Checks for power and ground shorts between two geometries belonging to different nets. The command performs power and ground short check between the following:

- PG and PG nets
- PG and signal nets
- PG and other special net

This command enables you to check for power and ground shorts in the design before performing any kind of power-grid analysis. You can use this command after restoring the physical design (`read_design -physical_data *.enc` or `read_def filename`).

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `check_pg_shorts` parameter. For a detailed description of the command and all its parameters, use the `man` command: `man check_pg_shorts`
`-area {x1 y1 x2 y2}`	Specifies the coordinates of the area to verify. For example, `check_pg_shorts -area {0 0 100 100}`. The unit of coordinates is micron. *Data_type*: rect, optional
`-net net_name`	Specifies the name of the power or ground net to be verified. *Data_type*: string, optional
`-no_cell_blockages`	Ignores cell blockages during during power and ground short checking. *Data_type*: bool, optional
`-no_route_blockages`	Ignores routing blockages during during power and ground short checking. *Data_type*: bool, optional
`-out_file filename`	Specifies the name of the report file that contains the violation information. *Data_type*: string, optional

Example

The following command checks for power and ground shorts in the area specified and generates a detailed report named `shorts.rpt`.

```
check_pg_shorts -area {0 0 1600 1600} -out_file shorts.rpt
```

Related Information

- `read_def`
- The Identifying and Viewing Violations chapter in the *Innovus User Guide.*

check_power_vias

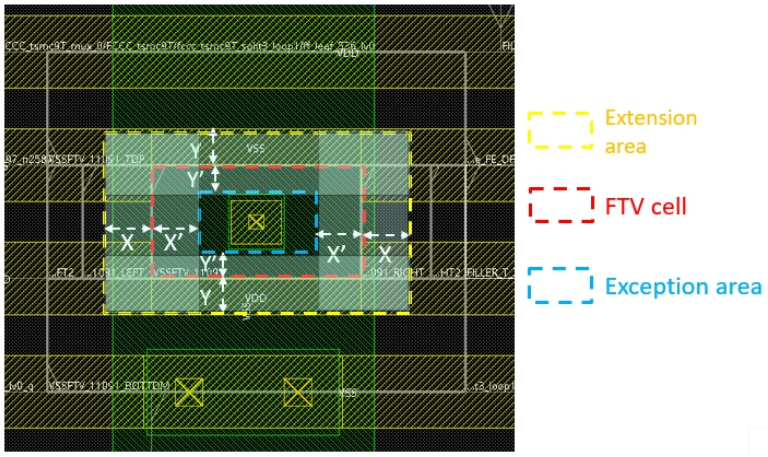
```
check_power_vias
[-help]
[-append]
[-area {x1 y1 x2 y2}]
[-check_wire_pin_overlap]
[-check_fill]
[-edge_to_edge]
[-error integer]
[-exclude_region_file check_file]
[-hookup_pitch pitch_value]
[-ignore_objects {drc_fill | io_wire}]
[-layer_range {bottomLayer topLayer}]
[-rail_layers layer_name]
[-stripe_layers layer_list]
[-nets {netNames} | -selected]
[-non_orthogonal_check [-non_orthogonal_distance {x y}]]
[-pitch pitch_value]
[-report file]
[-search_range {x y}]
[-shielding]
[-stacked_via]
[-stripe_rule value]
[-cut_area_ratio value]
[-what_if_report filename]
[-width_range string]
```

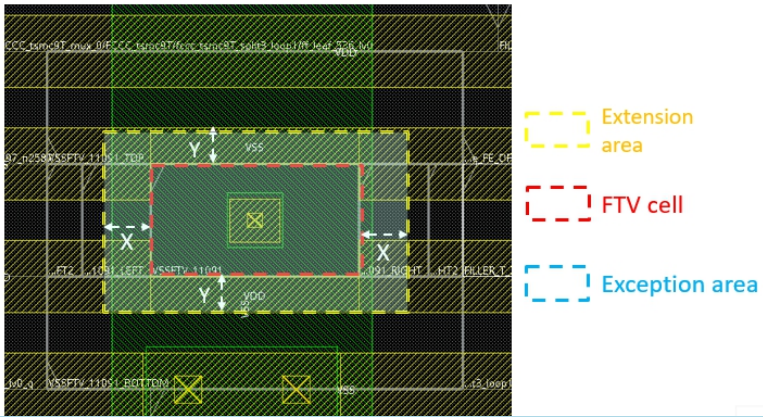
This command has a variety of power-rail overlap checks to look for missing power-grid vias. By default, it checks that orthogonal power-routes on adjacent routing layers have a via between them at every intersection. For example, there should be a via at the point where a Metal3 power stripe overlaps a Metal2 power stripe. By default, horizontal routes get checked only for overlaps with vertical routes. If the power shapes are polygon, the polygon is split into rectangles and the rectangles are used to decide the direction. Support for checking stacked vias between non-adjacent routing layers is optional. Violations are highlighted in the layout window, and a text report is generated with the location of the missing vias.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `check_power_vias` parameter. For a detailed description of the command and all its parameters, use the `man` command: `man check_power_vias`
`-append`	Specifies that all previous `check_power_vias` settings will be kept. *Data_type*: bool, optional
`-area x1 y1 x2 y2`	Specifies the coordinates of the area to be checked. *Data_type*: rect, optional
`-check_fill`	Specifies that tied-off metal fill should also be checked. Floating fill is still ignored. *Default*: Metal fill will be ignored. *Data_type*: bool, optional
`-check_wire_pin_overlap`	Specifies that wire overlaps of pin shapes should also be checked. By default, only wire overlaps with wires are checked. *Data_type*: bool, optional
`-cut_area_ratio value`	Checks if the cut area in the metal intersection area is sufficient. `viaUtil = Total cut area / Metal intersect area` If the actual ratio is less than specified ratio, `check_power_vias` marks it as violation. By default, the metal intersection area is the metal overlap area. If `setAddStripeMode -via_using_exact_crossover_size` is set to `false`, the partial overlapped metal will be extended to full width intersection area. *Data_type*: float, optional

`-edge_to_edge`	Checks the non-orthogonal distance from edge to edge within a layer. *Data_type*: bool, optional
`-error` *integer*	Specifies the maximum number of errors to report. *Data_type*: int, optional

`-exclude_region_file` *check_file*	Ignores missing VIA checks within the specified `check_file` regions for a cell extension area. The `check_file` format can be a set of the following values:
	• `cell`: Specifies the cell type.
	• `ext_layer_bottom`: Specifies the extension area bottom layer.
	• `ext_layer_top`: Specifies the extension area top layer.
	• `ext_x`: Specifies the extension length in the X direction.
	• `ext_y`: Specifies the extension length in the X direction.
	• `exc_x`: Specifies the exception length in the X direction.
	• `exc_y`: Specifies the exception length in the Y direction.
	Example 1:
	The following set of `check_file` formats ignores missing VIA check within the yellow rectangle (in the output), but checks missing VIA within the blue rectangle:
	`FTV_CELL, B_M0, B_M1, X, Y, X', Y'`
	`INV_D2, M0, M1, X2, Y2, X2', Y2'`
	Following is the output:
	 
	Example 2:
	The following set of `check_file` formats ignores missing VIA check within the yellow rectangle (in the output):
	`FTV_CELL, B_M0, B_M1, X, Y`
	`INV_D2, M0, M1, X2, Y2`
	Following is the output:

	
`-hookup_pitch` *pitch_value*	Specifies the distance between two vias, which are on the filler instance and are greater than or equal to the given value. *Data_type*: int, optional
`-ignore_objects {drc_fill | io_wire}`	Skips checking the power vias of the specified wire type. *Data_type*: (enum)+, optional

`-layer_range {`*bottomLayer topLayer*`}`	Checks for the missing stacked vias between the bottom and top layers. It also checks for the missing vias between all the adjacent intermediate layers, but does not check for the stacked vias between the intermediate layers. While using this parameter, you can specify full layer names like `METAL1`, or abbreviations like `1`. For example, `check_power_vias -layer_range {2 5}` checks for missing stacked vias between overlapping `metal2` and `metal5`, and also missing vias for overlapping `2/3`, `3/4`, `4/5`.
	This parameter checks the missing adjacent vias if `bottomLayer` is specified, but `topLayer` is not, with considering that `topLayer` is just one layer above. To review the missing stacked via or missing adjacent via DRC marker, use the `gui_show_markers` command for filtering the DRC marker. For example:
	1. To display only the missing stacked via DRC markers, run:
	`gui_show_markers -all -filter {stacked via} -search_description -filter_mode AND`
	2. To display only the missing via DRC markers between the adjacent layers, run:
	`gui_show_markers -all -filter {adjacent via} -search_description -filter_mode AND`
	See `gui_show_markers` for more information on the command usage.
	Data_type: string, optional

`-nets {`*netNames*`}`	Specifies the names of the power nets that will be checked. A single net or a list of nets enclosed in curly braces({}) or quotes (" ") can be specified. Wildcards (* and ?) are supported. For example, when you specify `check_power_vias -nets {v*}`, all PG net names having initial letter as `v` are checked. **Note:** Currently, wildcards are supported for checking PG nets only. *Default*: All power grid nets. *Data_type*: string, optional
`-non_orthogonal_check`	Checks the overlap area for non-orthogonal crossing wires (e.g. wires that are both in the same direction) in adjacent as well as non-adjacent layers. Checks for missing vias between two or more parallel wires. The via center-to-center distance between parallel wires should be less than or equal to the specified spacing If this parameter is not specified, horizontal wires get checked only against vertical wires. **Note**: You can use `-non_orthogonal_distance {x y}` parameter with `-non_orthogonal_check` to find missing vias within a specified window. When the `x` or `y` distance is provided, `check_power_vias` checks if the wire is parallel with another wire. If yes, it checks the via center-center spacing and via center-wire end distance between parallel wires. If the actual distance is greater than the required distance, it marks a violation. *Data_type*: bool, optional

`-non_orthogonal_distance {x y}`	Specifies distances for finding missing via within a specified window for a non-orthogonal check, with `x` and `y` representing the distance in the horizontal and vertical directions, respectively. When the `x` or `y` distance is provided, `check_power_vias` checks if the wire is parallel with another wire. If yes, it checks: • The via center to via center spacing between parallel wires against the value *x (y)* • The via center to wire end distance against the value *x/2 (y/2)* If the actual distance is greater than the required distance in either case, it marks a violation. When the `x` or `y` distance is not provided or set to `0`, the via to via distances are not checked. **Note**: This parameter can only be specified with the `-non_orthogonal_check` parameter. *Data_type*: point, optional
`-pitch pitch_value`	Specifies the distance between two vias, which are not on the filler instance and are greater than or equal to the given value. *Data_type*: int, optional
`-rail_layers layer_name`	Specifies a single power rail layer for missing via (only the via above power rail) check. The check depends on the `-stripe_rule` option. When `-stripe_rule` is specified, the via to via distance should be less than or equal to the specified value. If a distance value is not specified with the `-stripe_rule` option, the check is ignored. See the examples below. The `-rail_layers` option is not controlled by `-layer_range`. You can specify a power rail layer other than that specified with `-layer_range` for checks against the `stripe_rule` and `search_range` rules. *Data_type*: string, optional
`-report file`	Specifies the name of the output file for the report. *Data_type*: string, optional
`-search_range {x y}`	Specifies a rectangle area with `XY`. There should be one valid connection from power rail to power stripe in the specified area. By default, the valid connection would be searched from the uppermost power rail layer to all the power stripe layers. You can use the `-rail_layers` option to specify the power rail layer, and the `-stripe_layers` option to limit the power stripe layers. With the `-rail_layers` and `-stripe_layers` options, only the specified power rail and power stripes layers are used for valid connection check. The x and y values are floats in units of microns. Refer to the **Examples** section for search range examples. *Data_type*: point, optional
`-selected`	Checks for the missing vias between the selected PG wires and other intersecting selected/unselected PG wires. When this parameter is used with the `-check_wire_pin_overlap` parameter, it checks for the missing vias between the intersection of selected PG wires and pins of the selected instances. This parameter eliminates the requirement of running the command on the entire design, and hence improves the run time. *Data_type*: point, optional
`-shielding`	Specifies that shielding wires should also be checked. By default, shielding wires are not checked. *Data_type*: bool, optional
`-stacked_via`	Checks for missing vias between all non-adjacent as well as adjacent layers. *Data_type*: bool, optional
`-stripe_layers layer_list`	Specifies the power stripe layers to be checked for a valid connection path from power rail to power stripe. The check depends on the `-search_range` option. If `-search_range` is specified, at least one valid connection path should be found from the power rail to power stripe in that search range. If a search range is not specified with the `-search_range` option, the check is ignored. You can specify the power rail layer with the `-rail_layers` option. By default, only the uppermost power rail layer is checked. See the examples below. The `-stripe_layers` option is not controlled by `-layer_range`. You can specify power stripe layers other than that specified with `-layer_range` for checks against the `search_range` rule. *Data_type*: string, optional

`-stripe_rule` *value*	Specifies in microns the distance for power via. Only the power via above the power rail is checked. If the via to via center-to-center spacing is greater than the specified value, a violation marker is created. You can specify the power rail layer with `-rail_layers` for checking. By default, only the power via above the uppermost power rail is checked against the rule. Refer to the **Examples** section for stripe rule examples. *Data_type*: float, optional
`-what_if_report` *filename*	
	Writes out `check_power_vias` violations data into a ECO file that can be used directly as input for whatIf rail analysis. When you specify this parameter, the missing via information generated can be used as input for what-if virtual wires. *Data_type*: string, optional
`-width_range` *string*	
	Specifies the metal width to be considered. `check_power_vias` uses the specified width range to filter wires. Only the bottom and top wires that fall in the range are checked. The *`string`* value takes the following format: `-width_range {`*`minWidth`* `maxWidth [layerName …] [`*`minWidth maxWidth [layerName …]`*`] … }` • If *`minWidth`* is 0, all wires with width less than specified *`maxWidth`* will be checked. • If *`maxWidth`* is 0, all wires with width greater than specified *`minWidth`* will be checked. • If *`layerName`* is specified in the string, `widthRange` is enabled only for the specified layer. • If no *`layerName`* is specified in the string, `widthRange` is enabled for all non-specified layers. Refer to the **Examples** section for width range examples. **Note:** Use braces or quotation marks to enclose the string value for `widthRange`. *Data_type*: string, optional

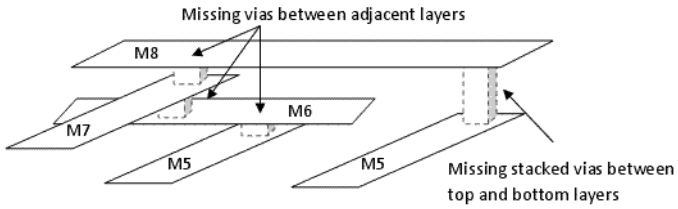
Examples

- The following command specifies that all vias for all powergrid nets will be checked, except for metal fill and shielding wires, and reports it to a file called `power_via.report`:

  ```
  check_power_vias -report power_via.report
  ```

- The following command checks for missing stacked vias between `METAL5` and `METAL8` layer intersections on the VSS net. In addition, it checks for missing vias between `METAL5-METAL6`, `METAL6-METAL7` and `METAL7-METAL8` layer intersections. The violations are reported in the `powerVia.rpt` file and highlighted in GUI.

  ```
  check_power_vias -layerRange {METAL5 METAL8} -nets VSS -report powerVia.rpt
  ```



- The following command checks that the overlapping power routes on adjacent layers have vias that cover at least 70% of the overlap area..

  ```
  check_power_vias -viaUtil 0.7
  ```

- The following command checks that the overlapping power routes have vias that cover at least 70% of the overlap area for layers between `METAL3` and `METAL4`.

  ```
  check_power_vias -viaUtil 0.7 -layer_range {METAL3 METAL4}
  ```

- The following command checks for wires overlaps for all pin shapes included, enclosed, or touched by the specified area:

  ```
  check_power_vias -area {100 100 1000 1000} -check_wire_pin_overlap
  ```

- The following command checks wires with width greater than `0.032` and less than `0.096` for all layers:

  ```
  check_power_vias -width_range {0.032 0.096}
  ```

- The following command checks wires with width in the `0.032` ~ `0.096` range for metal1/metal2 and width `0.038~0.114` for metal3/metal4. For other layers, there is no width control:

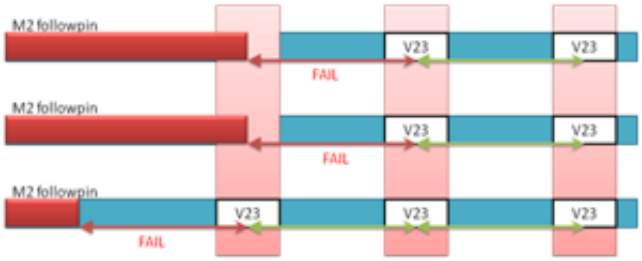
  ```
  check_power_vias -width_range {0.032 0.096 metal1 metal2 0.038 0.114 metal3 metal4}
  ```

- The following command checks wires with width in the `0.032` ~ `0.096` range for metal1/metal2. For other layers, it checks wires with width in the `0.038~0.114` range:

  ```
  check_power_vias -width_range {0.032 0.096 metal1 metal2 0.038 0.114}
  ```

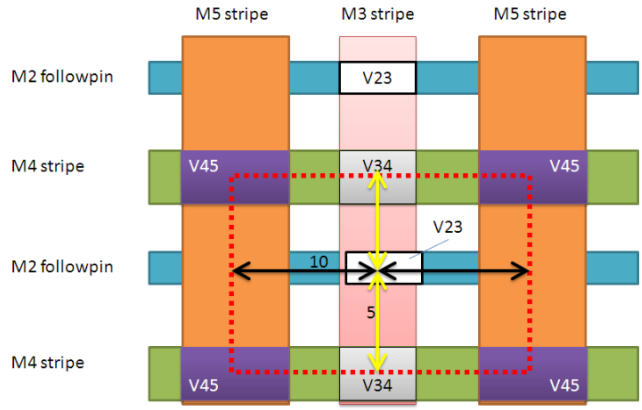
- The following command checks the via between power rail layer metal`2` and metal`3`. If the via's center-to-center distance is more than `17u`, the tool reports a violation and places a marker on the power rail wires.

  check_power_vias -stripe_rule 17 -rail_layers metal2

- The following command checks the valid connection for metal`2` power rail to stripe layer metal`4` and metal`5` with the search rectangle area `10*5`. In the diagram below, there is no violation as the connection is built from metal`2` to metal`4`. If the `V34` vias are removed, then there is a violation as the connection cannot be built from metal`2` to metal`4`.

  check_power_vias -search_range {10 5} -rail_layers metal2 -stripe_layers {metal4 metal5}

Related Information

Prior to Running Early Rail Analysis section in the Power and Rail Analysis chapter of the *Innovus User Guide*

check_process_antenna

```
check_process_antenna
[-help]
[-detailed]
[-error value]
[-max_floating_area_different_net]
[-nets {netNames} | -selected]
[-no_io_pin_default]
[-no_max_float_area]
[-out_file filename]
[-pg_net]
```

Verifies process antenna effect (PAE) and maximum floating area violations.

Before running this command, make sure that process antenna or maximum floating area keywords are specified in the LEF file and the signal nets are routed.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `check_process_antenna` parameter. For a detailed description of the command and all its parameters, use the `man` command: `man check_process_antenna`
`-detailed`	Generates a report containing the specified information for all nets, including those with no violations. *Data_type*: bool, optional
`-error value`	Specifies the maximum number of error messages to report. *Default:* `1000` *Data_type*: int, optional
`-max_floating_area_different_net`	Verifies the maximum floating area between different nets. *Data_type*: bool, optional
`-nets {netNames} | -selected`	Reports named or selected nets. Select nets by using the select_obj command. If you specify one net name, you do not need the braces around the name. *Data_type*: string, optional
`-no_io_pin_default`	Specifies that the following LEF keywords are not applied to the I/O pins: • `ANTENNAINPUTGATEAREA` • `ANTENNAINOUTDIFFAREA` • `ANTENNAOUTPUTDIFFAREA` *Data_type*: bool, optional
`-no_max_float_area`	Disables check for maximum floating area violations. *Data_type*: bool, optional
`-out_file filename`	Specifies the report file. The file can contain both PAE and maximum floating area violation information. *Default:* `design.antenna.rpt` *Data_type*: string, optional
`-pg_net`	Checks tie-high and tie-low nets for process antenna violations. *Data_type*: bool, optional
`-selected`	Specifies whether to check the selected nets. *Data_type*: bool, optional

Example

The following command verifies PAE violations in the routed design and generates a report file called `Test7.antenna.rpt`:

```
check_process_antenna -out_file Test7.antenna.rpt
```

Related Information

- The following sections in the Identifying and Viewing Violations chapter in the *Innovus User Guide*
  - Verifying Process Antennas
  - Verifying Well-Process-Antenna Violations
  - Verifying Maximum Floating Area Violations
- The Repairing Process Antenna Violations section in the Using the NanoRoute Router chapter in the *Innovus User Guide*
- The following sections in the *LEF/DEF Language Reference* document:
  - Calculating and Fixing Process Antenna Violations
  - LEF Syntax

check_tieoffs

```
check_tieoffs
[-help]
[-no_tieoff_pins_file filename]
[-out_file filename]
[-power_domain string]
```

Checks tie cell connections. `check_tieoffs` honors the tie high/low cell list from `set_db -add_tieoffs_cells` and checks whether or not the tie high/low nets are connected to the tie high/low cells.

Use this command after using `add_tieoffs` to add instances of tie-off cells to the logical hierarchy of the design.

The expected flow is as follows:

1. Specify the tie cell names and other settings with the `add_tieoffs` attributes.
2. Add tie high/low cells with `add_tieoffs`.
3. Run `check_tieoffs` to check whether tie cell connections are correct.

Violations can be viewed in Violation Browser.

Parameters

`-help`	Prints a brief description that includes the type and default information for each `check_tieoffs` parameter. For a detailed description of the command and all its parameters, use the `man` command: `man check_tieoffs`
`-no_tieoff_pins_file` *`filename`*	Specifies a file containing exception list of instance/pin names. Use this parameter to specify the instance/pins that should not have any tie cell connections. The file should list instance and pin names in the following format: *`instance_name/pin_name`* *Data_type*: string, optional
`-out_file` *`filename`*	Specifies the report file for recording tie cell violations. By default, the report is saved as *`designName`*`.tie.rpt`. *Data_type*: string, optional
`-power_domain` *`string`*	Specifies the power domain name to be checked. *Data_type*: string, optional

Related Information

- `add_tieoffs`
- The Verifying Tie Cells section in the Identifying and Viewing Violations chapter of the *Innovus User Guide*

check_well_antenna

```
check_well_antenna
[-help]
[-need_protect_cells {cell_list}]
[-protect_cells {cell_list}]
[-not_protect_cells {cell_list}]
[-report filename [-detailed]]
```

Checks for any CORE rows that have well-process-antenna violations. Well-process-antenna violations are normally caused by rows with decap cells that do not have any protecting cells in the same row. This command marks with a violation marker any such CORE cell. You can then write a Tcl script to put in a well-antenna cell next to the violation marker.

Parameters

-help	Prints a brief description that includes the type and default information for each check_well_antenna parameter. For a detailed description of the command and all its parameters, use the man command: man check_well_antenna
-detailed	Reports all [CORE | ENDCAP] cells' protect information in a report file. *Data_type*: bool, optional
-need_protect_cells {*cell_list*}	Specifies the cell names that need protection (typically decap cells). This option also indicates that the specified cells do not protect the row they are in. Pattern matching with * is allowed in cell names. *Data_type*: string, optional
-not_protect_cells {*cell_list*}	Specifies the cell names that are to be marked as cells that do not protect rows. By default, the cells with LEF class CORE but without a subclass (e.g. normal BUF and AND2 cells) are assumed to have a well discharge path that protects a row from well-antenna violations. Use -not_protect_cells option to override the default for any of these cells that do not protect a row. Pattern matching with * is allowed in cell names. *Data_type*: string, optional
-protect_cells {*cell_list*}	Specifies the cell names that are to be marked as protection cells. By default, cells with CLASS CORE and a subclass {FEEDTHRU | TIEHIGH | TIELOW | SPACER | ANTENNACELL | WELLTAP} or cells with class ENDCAP (with or without any subclass) do not have a well-discharge path to protect a row. Use the -protect_cells option to override the default for any of these cells that do protect the row. Pattern matching with * is allowed. *Data_type*: string, optional
-report *filename* [-detailed]	Specifies the report file name. By default, the report includes information about only the cells that you specified with the options. If you specify -detailed, check_well_antenna reports every class CORE and ENDCAP cell with its subclass and protection status to the *filename* report. *Data_type*: string, optional

Examples

Normally, the decap cells are the only cells that need protection. However, if the standard-cell library in your design has one of the ENDCAP cells with a well-antenna protection device built-in, and the filler, well-tap, tie-high and tie-low cells are just class CORE rather than with the correct CORE sub-class, you can use the following command:

```
check_well_antenna -need_protect_cells decap* -protect_cells wellAntEndCap -not_protect_cells {filler* welltap* tie*} -report wellant.rpt
```

Related Information

The 'Verifying Well-Process-Antenna Violations' section in the Identifying and Viewing Violations chapter in the *Innovus User Guide.*

check_well_taps

```
check_well_taps
[-help]
[-area x1 y1 x2 y2]
[-avoid_abutment]
[-cells list_of_well_tap_cells]
[-from_edge]
[-max_distance distance]
[-layer layer_name]
[-power_domain powerDomainName]
[-report filename]
[-site_offset number_of_sites]
[-well_cut_cells list_of_well_cut_cells]
[-well_layer string]
```

Generates violation markers for missing well-tap cells and for well-tap cells that do not meet the rule specified for the distance between well-tap cells. Optionally, generates a well-tap violation report.

Honors the `set_well_tap_mode` settings for `-cells`, `-max_distance`, `-well_cut_cell`, `-bottom_tap_cell`, and `-top_tap_cell` options.

Marks violations in the area where well-tap placement violates the rule, covering the shared well of abutting rows. If there is no abutting row, marks the violation where it covers only one-half of a row with a violation.

Reports the actual distance between well-tap cells, as well as the rule distance. If there are no well-tap cells in a row and the row that abuts it, or no well-tap cells between the row end and the edge of a block, the software generates a message.

Does not check row areas that are covered by blocks and their halos, or by placement blockages.

The software places well-tap cells with the `add_well_taps` command. You can delete well-tap cells with the `delete_filler` command.

Parameters

`-help`	Prints a brief description that includes the type and default information for each `check_well_taps` parameter. For a detailed description of the command and all its parameters, use the `man` command: `man check_well_taps`
`-area x1 y1 x2 y2`	Specifies the area in which well-taps are to be checked. *Data_type*: rect, optional
`-avoid_abutment`	Checks and ensures that there is no abutted tap cell in the vertical and horizontal directions. *Data_type*: bool, optional
`-cells list_of_well_tap_cells`	Specifies the well-tap cells. If not specified, uses cells defined as `CLASS CORE WELLTAP` in the Macro section of the LEF file. *Data_type*: string, optional
`-from_edge`	Checks the rule distance from the right and left edges of the well tap cells, instead of from the center line. Use this parameter when the design has well-tap cells of different widths. If this parameter is not specified, `check_well_taps` checks the rule distance from the center line of the well-tap cells. *Data_type*: bool, optional
`-layer layer_name`	Specifies the layer name to check the rule. *Data_type*: string, optional
`-max_distance distance`	Specifies the maximum distance between two well-tap cells. The spacing between two well-tap cells should be twice the `-max_distance` value when measured from center to center or edge to edge. The distance between tap and row end should be same as the `-max_distance` value. For example, `check_well_taps -max_distance 50`. *Data_type*: float, optional
`-power_domain powerDomainName`	Limits checking to the specified power domain. If you do not use this parameter, *`check_well_taps`* checks the default domain with the designs of N3 or below nodes, and checks the entire design for the other nodes. *Data_type*: string, optional
`-report filename`	Specifies the report name. *Default:* *`topCellName`*`.welltap.rpt` *Data_type*: string, optional
`-site_offset number_of_sites`	Checks the offset value in number of sites that vertical taps need to be aware of. *Minimum*:`1` *Maximum*: `INT_MAX` *Data_type*: int, optional
`-well_cut_cells list_of_well_cut_cells`	Specifies the list of cells with the well cuts. Use this option to flag latch-up violations due to well cut cells. *Data_type*: string, optional
`-well_layer string`	Checks the latchup rule for `pwell` or `nwell`. *Data_type*: string, optional

Examples

The following command displays violations due to well cut cells:

```
check_well_taps -max_distance 30 -well_cut_cells {EP32_VDDI_DSWITCH_50n_NOBUF EP32_VDDI_DSWITCH_50n_BUF} -cells {EP8T32_VDDI_GNDS_CDM EP8T32_VDDI_VDDS_CDM}
```

Related Information

- The following sections in the Placing the Design chapter of the *Innovus User Guide*:
  - Adding Well-Tap Cells
  - Deleting Well-Tap Cells
- The Substrate Biasing section in the Low Power Design chapter of the *Innovus User Guide*
- ECO Flows chapter of the *Innovus User Guide*

check_wire_gaps

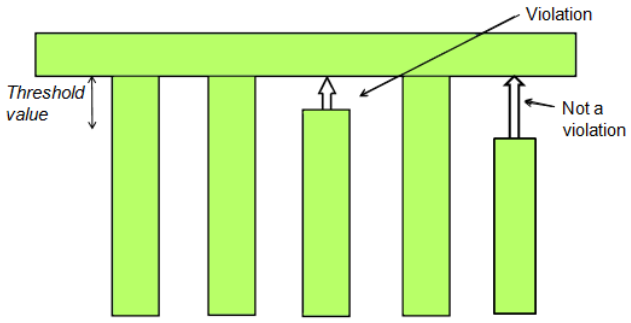
```
check_wire_gaps
[-help]
[-append]
[-area {x1 y1 x2 y2}]
[-layers layer_names]
[-nets power_nets]
[-out_file filename]
[-wire_to_boundary [-design {v1 v2}] [-core {v1 v2}] [-block {v1 v2}] [-power_domain {v1 v2}]]
[-wire_to_shape [-ring value]]
[-wire_to_wire value]
```

Verifies the continuity of follow pin and stripes. Use this command to identify power strap failures due to broken power wires in the power planning stage. The `check_wire_gaps` command gives you the flexibility to check for breaks in a specific area and for specific layers or power nets.

Additionally, you can use this command to ensure that wires do not stop short of the expected distance from the chip, core, or block boundary.

Parameters

`-help`	Prints a brief description that includes the type and default information for each `check_wire_gaps` parameter. For a detailed description of the command and all its parameters, use the `man` command: `man check_wire_gaps`
`-append`	Specifies that all previous `check_wire_gaps` settings will be kept. *Data_type*: bool, optional
`-area {x1 y1 x2 y2}`	Specifies the area to be checked. *Data_type*:
`-block {v1 v2}`	Specifies the wire-to-block gap distance. Here: • `v1`: Specifies the threshold value or acceptable distance from the block boundary to the wire. • `v2`: Specifies how far to look from the block boundary to find wires that should be considered for the check. Use this check to ensure that vertical straps cover the horizontal straps underneath them near a block edge. For example, if you specify `check_wire_gaps -wire_to_bounday -block {0 0.3}`, any wire that is at a distance less than 0.3 um from the block boundary is reported as a violation. Wires that are at a greater distance than 0.3um are not flagged as violations as such breaks are assumed to be intentional. *Data_type*: point, optional
`-core {v1 v2}`	Specifies the wire-to-core boundary gap distance. Here: • `v1`: Specifies the threshold value or acceptable distance from the core boundary to the wire. • `v2`: Specifies how far to look from the core boundary to find wires that should be considered for the check. For example, if you specify `check_wire_gaps -wire_to_bounday -core {0 0.3}`, any wire that is at a distance less than 0.3 um from the core boundary is reported as a violation. Wires that are at a greater distance than 0.3 um are not flagged as violations as such breaks are assumed to be intentional. *Data_type*: point, optional
`-design {v1 v2}`	Specifies the wire-to-design boundary gap distance. Here: • `v1`: Specifies the threshold value or acceptable distance from the chip boundary to the wire. • `v2`: Specifies how far to look from the chip boundary to find wires that should be considered for the check. For example, if you specify `check_wire_gaps -wire_to_bounday -design {0 0.3}`, any wire that is at a distance less than 0.3 um from the chip boundary is reported as a violation. Wires that are at a greater distance than 0.3 um are not flagged as violations as such breaks are assumed to be intentional. *Data_type*: point, optional
`-layers layer_names`	Specifies the metal layers to be considered. *Data_type*: string, optional

`-nets power_nets`	Specifies the name of the power nets that will be checked. *Data_type*: string, optional
`-out_file filename`	Specifies the report file for `check_wire_gaps` violations data. *Data_type*: string, optional
`-power_domain {v1 v2}`	Specifies the wire-to-power-domain gap distance. Here: • `v1`: Specifies the threshold value or acceptable distance from the power domain to the wire. • `v2`: Specifies how far to look from the power domain to find wires that should be considered for the check. *Data_type*: point, optional
`-ring value`	Specifies the wire-to-ring gap distance. *Data_type*: float, optional
`-wire_to_boundary`	Checks the wire to boundary gap. You must specify one of the following sub-options with `-wire_to_boundary`: • `-block` to check the gap between wire and block boundary. • `-design` to check the gap between wire and chip boundary. • `-core` to check the gap between wire and core boundary. • `-power_domain` to check the gap between wire and power domain. *Data_type*: bool, optional
`-wire_to_shape`	Checks the gap between the wire and some specific shapes. *Data_type*: bool, optional
`-wire_to_wire value`	Specifies the wire-to-wire gap distance. Use this check to ensure that wires do not stop short of another wire of the same net or layer. For instance, you can use this check to ensure that wires connect properly to a power ring. If the wire-to-wire gap in the same net is less than the specified threshold value, `check_wire_gaps` flags it as a violation. If the wire-to-wire is greater than the threshold value, the break is assumed to be intentional.  *Data_type*: float, optional

Example

The following command checks wire-to-wire gap and flags wires in the same net that are closer than 0.2 um:

```
check_wire_gaps -wire_to_wire 0.2
```

Related Information

The Identifying and Viewing Violations chapter in the *Innovus User Guide.*

create_marker

```
create_marker
[-help]
{-bbox {x1 y1 x2 y2} | -poly {x1 y1 x2 y2...}}
[-description description]
[-layer layerName]
[-rule_map]
[-sub_type subtypeName]
[-tool toolName]
[-type typeName]
```

Creates markers for violations in the database and imports DRC markers generated by other software applications. Use this command if you want to import only a specific marker or check a specific marker that has been created by a third party tool.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `create_marker` parameter. For a detailed description of the command and all its parameters, use the `man` command: `man create_marker`
`-bbox x1 y1 x2 y2`	Specifies the area of the bounding box of the marker. For example, `create_marker -bbox 100 100 100.5 100.5`. *Data_type*: string, optional
`-description description`	Describes the marker. For example, `create_marker -description "the user mark is non default rule spacing check"`. *Default:* `""` *Data_type*: string, optional
`-layer layerName`	Specifies the layer. For example, `create_marker -layer METAL2`. *Data_type*: string, optional
`-poly {x1 y1 x2 y2...}`	Specifies the area of the bounding polygon of the marker. *Data_type*: string, optional
`-rule_map`	Specifies if the marker needs to be created through `rule_map`. *Data_type*: bool, optional
`-sub_type subtypeName`	Specifies the marker subtype. For example, `create_marker -sub_type user_spacing`. *Default:* `Other` *Data_type*: string, optional
`-tool toolName`	Specifies the source software application. For example, `create_marker -tool Innovus`. *Default:* `Other` *Data_type*: string, optional
`-type typeName`	Specifies the marker type. For example, `create_marker -type user_verify`. *Default:* `Other` *Data_type*: string, optional

Example

- The following command generates markers for tool named `Calibre` with type `METAL1` and subtype `S.1`:

  ```
  create_marker -bbox { 1 2 1 2 } -tool Calibre -type METAL1 -sub_type S.1
  ```

  The violation browser displays the marker in the above example as follows:

  ```
  + Calibre
      + METAL1
          + S.1
              ...
  ```

- Suppose the cadence LPA tool has created the following marker on layer `METAL3`:

  ```
  CDNLitho;SPACING; Severity: 3; MARKER_INFO: "CD = 45.39"

  bbox = (468.248, 63.511) (468.310, 63.599)
  ```

  The following command creates a corresponding marker in Innovus design:

  ```
  create_marker -bbox {468.248 63.511 468.310 63.599} -layer metal3 /
              -type CDNLitho -sub_type Spacing
  ```

Related Information

- `delete_markers`
- Identifying and Viewing Violations chapter in the *Innovus User Guide*

delete_drc_markers

```
delete_drc_markers [-help]
```

Clears all design rule checking (DRC) markers in your design.

Note: Do not use the `delete_drc_markers` command before using `ecoRoute` or any other routing command. This is because NanoRoute works only on the existing DRC/violation markers. The routing commands will not necessarily perform DRC checks in the routing halo regions if the violation markers are already cleared.

Parameters

`-help`	Prints a brief description of the `delete_drc_markers` command usage. For a detailed description of the command, use the `man` command: `man delete_drc_markers`

Related Information

- `check_drc`
- Identifying and Viewing Violations chapter in the *Innovus User Guide*

delete_markers

```
delete_markers
[-help]
[-all]
[-area {x1 y1 x2 y2}]
```

Deletes markers from a specified area or the entire design area.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `delete_markers` parameter. For a detailed description of the command and all its parameters, use the `man` command: `man delete_markers`
`-all`	Deletes markers from the entire design area. *Data_type*: bool, optional
`-area {x1 y1 x2 y2}`	Specifies a bounding box for the area from which to delete markers. For example, `delete_markers -area {0 0 100 100}`. *Data_type*: string, optional

Related Information

The following chapters in the *Innovus User Guide*:

- `create_marker`
- Identifying and Viewing Violations
- Using the NanoRoute Router - Evaluating Violations section

get_signoff_verify_design_config

```
  get_signoff_verify_design_config
[-help]
[-abort_on_layout_error]
[-abort_on_missing_rulecheck]
[-area]
[-attach_instance_name]
[-attach_net_name]
[-attach_net_prop]
[-auto_merge_base_class]
[-config_file]
[-control_file
[-core_license]
[-die_area_as_boundary]
[-distributed]
[-distributed_timeout]
[-error_limit]
[-extra_options]
[-format]
[-ignore_blockage]
[-ignore_fill]
[-keep_data]
[-layer_map_file]
[-layer_range]
[-layers]
[-layout_path]
[-lib_name]
[-license_distributed_continue]
[-license_stacking]
[-license_timeout]
[-merge]
[-merge_path]
[-mode]
[-multi_cpu]
[-multi_thread]
[-net_map_file]
[-nets]
[-no_structure_name]
[-offset]
[-output_macros]
[-pegasus_bin]
[-process_node]
[-pvs_fill]
[-report_file]
[-rule_file_drc]
[-rule_file_smart_verify_lvs]
[-rule_selection_options]
[-run_name]
[-selected_inst]
[-skip_auto_load_results]
[-stripes]
[-structure_name]
[-transform_to_inst_master]
[-ui_data]
[-uniquify_cell_names]
[-units]
[-working_dir]
```

Prints parameters for the `signoff_verify_design` command. It also prints `rule_selection_options` for a given process node that can be combined with `all {} | beol {} | feol {}`.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `get_signoff_verify_design_config` parameter. For a detailed description of the command and all its parameters, use the `man` command: `man get_signoff_verify_design_config`
`parameter_names`	Displays information for the specified parameters. You can specify one or more parameters. See `set_signoff_verify_design_config` for descriptions of the parameters you can specify.

Examples

- The following command displays the current setting of the `-stripes` parameter:
  `get_signoff_verify_design_config -stripes`
- The following command displays the current settings for all the `signoff_verify_design` parameters:
  `get_signoff_verify_design_config`

Related Information

- `set_signoff_verify_design_config`
- `signoff_verify_design`

gui_show_markers

```
gui_show_markers
[-help]
[-ac_limit]
[-all]
[-area {x1 y1 x2 y2}]
[-connectivity]
[-density]
[-display_by_layer]
[-geometry]
[-max_error_per_type value]
[-no_false_marker]
[-overlap]
[-process_antenna]
[-search_description]
[-short]
[-filter_query LO_formula | {-filter filterString -filter_mode {AND | OR | NOT}}]
```

Displays a list of violations. The software generates markers for the violations after you use one or more verification commands.

Geometry or Metal Density Violations

Updates geometry and metal density violation markers incrementally during an Innovus session--that is, it displays the markers generated the first time you run the `check_metal_density` command and adds new markers, or deletes markers, on subsequent runs during the same session. If the software finds violations during a subsequent run that were already found previously, the browser display does not change, as there is no incremental update.

The browser can make incremental changes because `check_metal_density` can check a small area of the design and update the database. As a result of this behavior, Innovus saves the information from the first verification run.

Connectivity, Process Antennas, or AC Limit Violations

Overwrites connectivity, process antenna, and AC limit violation markers if the `check_connectivity`, `check_process_antenna`, or `check_ac_limits` commands are run more than once during an Innovus session. These commands are net-based, not area-based, so the browser does not make incremental updates for them. As a result of this behavior, Innovus does not keep the information from the first verification run.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `gui_show_markers` parameter. For a detailed description of the command and all its parameters, use the `man` command: `man gui_show_markers`
`-ac_limit`	Displays AC limit violations. *Data_type*: bool, optional
`-all`	Displays all the violations that are present in the design. *Data_type*: bool, optional
`-area {x1 y1 x2 y2}`	Specifies the coordinates of the area whose violations to display. For example, `gui_show_markers -area {0 0 100 100}`. *Data_type*: string, optional
`-connectivity`	Displays connectivity violations. *Data_type*: bool, optional
`-density`	Displays metal density violations. *Data_type*: bool, optional
`-display_by_layer`	Displays the list of violations by the layer instead of route type. *Data_type*: bool, optional

`-filter filterString`	Specifies a text string in the violation message. Violation Browser searches the message of each violation and displays only the violations whose messages match the search conditions. To specify a list of strings, separate each string with a space. To specify a literal string with space in it, enclose the string in double quotation marks. For example, you can specify strings such as the following: • `METAL5` • `(600, 500)` • `METAL3 (700, 400)` • `"Pin of Net"` *Data_type*: string, optional
`-filter_mode {AND | OR | NOT}`	Specifies the condition for searching multiple strings in the violation browser. You cannot filter complex expressions because the `AND`, `OR`, and `NOT` parameters are mutually exclusive. • The `AND` and `OR` conditions work on single strings or multiple strings. • The `NOT` condition works only on a single string. *Data_type*: string, optional

`-filter_query LO_formula`	Specifies a complex logical expression for filtering violations. This option cannot be used along with `-filter` or `-filter_mode`. You can specify the AND, OR, and NOT conditions in the search expression. Use:
	• `&&` between strings in the expression to specify the AND condition.
	• `||` between strings in the expression to specify the OR condition.
	• ! before a string to specify the NOT condition. Note that ! is an one's complement operator. So for example, `!{object(net1) && layer(m2)}` is not allowed.
	The `-filter_query` parameter supports searches in the object, layer, and description areas. The area can be specified using the `object`, `layer`, and `desc` keywords. For example:
	`object(net1) || layer(m2) && desc(xxx)`
	You can also use the wildcards * and ? in the search expression. For example:
	`!object(clk*) && layer(m?)`
	To specify a literal string with space in it, enclose the string in double quotation marks. For example:
	`!object(clk*) && desc("prl spacing")`
	Data_type: string, optional

`-geometry`	Displays geometry violations. *Data_type*: bool, optional
`-max_error_per_type value`	Specifies a limit for the number of violations parsed for each violation type. For example, if you set `gui_show_markers -max_error_per_type 500`, the Violation Browser parses and displays only 500 markers per error type. Use this parameter if you are loading a large DRC file with millions of violations to prevent the database from becoming slow while loading the file and to limit the number of violations you review in the browser. By default, the Violation Browser displays **all** markers. *Data_type*: int, optional
`-no_false_marker`	Prevents Innovus from displaying false violations. *Data_type*: bool, optional
`-overlap`	Displays overlaps in the design. *Data_type*: bool, optional
`-process_antenna`	Displays process antenna violations. *Data_type*: bool, optional

`-search_description`	Searches for text in the violation message and the violation description. *Data_type*: bool, optional
`-short`	Displays short violations. *Data_type*: bool, optional

Examples

The following command displays all violations whose marker title contains `METAL3`:

```
gui_show_markers -all -filter METAL3
```

The following command displays geometry violations whose marker title contains `METAL3` or `tdsp`:

```
gui_show_markers -geometry -filter {METAL3 tdsp} -filter_mode OR
```

The following command displays geometry violations whose marker title or marker description field contains `Min` and `0.24` and `METAL3`:

```
gui_show_markers -geometry -filter {Min 0.24 METAL3} -filter_mode AND -search_description
```

The following command displays violations occurring on Metal4 or Metal5 layers but excludes short violations:

```
gui_show_markers -filter_query {!Short && (m4 || m5)}
```

Related Information

- `check_connectivity`
- `check_process_antenna`
- `check_ac_limits`
- The following sections in the Identifying and Viewing Violations chapter of the *Innovus User Guide*:
  - Viewing Violations With the Violation Browser
  - Clearing Violations
- The Evaluating Violations section in the Using the NanoRoute Router chapter of the *Innovus User Guide*

opt_pattern

```
opt_pattern
[-help]
[-back_out string]
[-check_function string]
[-check_local]
[-config string]
[-create]
[-deck string]
[-dir string]
[-env_layers string]
[-exclude string]
[-exclude_halo string]
[-extract]
[-extract_halo string]
[-extract_text_pattern]
[-find]
[-find_limit string]
[-fix]
[-fix_limit string]
[-guides_only]
[-include string]
[-incremental_check string]
[-input string]
[-layer_map_file string]
[-layers string]
[-map_file string]
[-marker_layers string]
[-markers string]
[-no_compress]
[-no_violation_browser]
[-oasis]
[-oasis_files string]
[-read]
[-selection string]
[-stream_files string]
[-techfile string]
[-write]
[-write_stream_options string]
```

Runs LPA on the selected design to flag Litho hotspots. It helps in creating, extracting, finding, and fixing patterns. This command uses squish and TVL patterns to find locations within a layout. Once a is pattern defined, you can use it for a variety of operations. Patterns can be extended to define via replacement, landing pad extensions, polygon insertions, and routing blockages. Patterns are commonly used to define markers for DRC or Litho hotspots.

Parameters

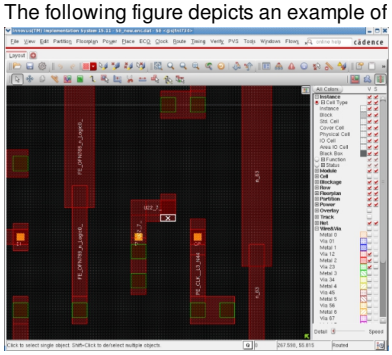
`-help`	Outputs a brief description that includes the type and default information for the `opt_pattern` parameters. For a detailed description of the command, use the man command: `man opt_pattern`
`-back_out string`	Reverts the fixing on the nets. *Data_type*: string, optional
`-check_function`	Specifies the DRC function. *Data_type*: string, optional
`-check_local`	Applies fixing with the local DRC check. *Data_type*: bool, optional
`-config string`	Specifies the name and path of the LPA custom configuration file. This file, if specified, controls all the run options of LPA. *Data_type*: string, optional

`-create`	Streams out the Innovus design into a GDS file and loads it in `design_review` or `k2_viewer`. ⚠ Specify the stream out map file with `-map_file`. *Data_type*: bool, optional
`-deck string`	Specifies the pattern deck to be used for LPA hotspot detection. ⚠ • The configuration variable, `lpaTechFile`, needs to be specified in the configuration file. • The LPA layer map needs to contain all the layers mentioned in the deck file. *Data_type*: string, optional
`-dir string`	Specifies the LPA output directory. This directory contains individual subdirectories for each layer. *Data_type*: string, optional
`-env_layers string`	Specifies the list of extra layers to be included in `write_stream` GDS/OASIS. ⚠ The layer number and data type of these extra layers need to be included in the LPA layer map. *Data_type*: string, optional
`-exclude string`	Specifies the list of nets that need not be used while fixing hotspots. *Data_type*: string, optional
`-exclude_halo string`	Specifies an extra halo around the marker to check the exclude nets. ⚠ This parameter works with `-exclude`. *Data_type*: string, optional
`-extract_text_pattern`	Produces squish patterns as opposed to TVL patterns, for extraction. *Data_type*: bool, optional
`-find`	Detects and fixes LPA hotspots at the time of run. *Data_type*: bool, optional
`-find_limit string`	Specifies the number of hotspots to be detected. *Data_type*: string, optional
`-fix`	Helps LPA in applying the hints to fix hotspots. *Data_type*: bool, optional
`-fix_limit string`	Specifies the number of hotspots to be fixed. *Data_type*: string, optional
`-guides_only`	Specifies LPA to turn off fixing of hotspots after detection. ⚠ The default behavior of the `-find` parameter is to find and fix hotspots. *Data_type*: bool, optional

`-include string`	Specifies the list of nets to be used while fixing hotspots. *Data_type*: string, optional
`-incremental_check string`	Specifies the name and path of the HIF file to be used for incremental validation. This HIF file includes the locations that identify the areas affected by each hotspot fix. LPA reads these locations and performs incremental checking only in these areas. This reduces the time required for validation. *Data_type*: string, optional
`-input string`	Specifies the path of a GDS. It opens design review with the given file. *Data_type*: string, optional
`-layer_map_file string`	Specifies the path of LPA layer map. ⚠ It can also be specified by including the following string in the configuration file: `setConf layerMapFile` *Data_type*: string, optional
`-layers string`	Specifies the list of layers to run on LPA. ⚠ It can also be specified by including the following string in the configuration file: `setConf layerOrder` *Data_type*: string, optional
`-map_file string`	Specifies the name and path of the Stream Out Map file created by Innovus to map the GDS stream to the layers in the Innovus database. If the GDS List file and Stream Out Map file are specified, LPA runs on the potential IP blocks defined in these files. *Data_type*: string, optional
`-marker_layers string`	Specifies the layers to be extracted in pattern extraction. ⚠ This parameter works along with `-markers`. *Data_type*: string, optional
`-markers string`	Specifies the path to a marker file in the HIF/RDB format for pattern extraction. *Data_type*: string, optional
`-no_compress`	Inputs GDS instead of `gds.gz`. *Data_type*: bool, optional
`-no_violation_browser`	Prevents LPA from loading the detected hotspots to the Innovus Violation Browser. *Data_type*: bool, optional
`-oasis`	Ensures that the stream out process uses OASIS output format. *Data_type*: bool, optional
`-oasis_files`	Specifies the name and path of the text file containing the list of OASIS files for LEF abstracts. *Data_type*: string, optional

`-read`	Loads guideline fixing from the database. *Data_type*: bool, optional
`-selection`	Specifies a list of hotspot type and severity to be extracted in the following format: `[list <type1>:<level1> <type2>:<level2> …]` *Data_type*: string, optional
`-stream_files string`	Specifies the name and path of the text file containing the list of GDS files for LEF abstracts. *Data_type*: string, optional
`-techfile`	Specifies the path of non-default LPA Mx technology file. *Data_type*: string, optional
`-write`	Saves guideline fixing from the database. *Data_type*: bool, optional
`-write_stream_options`	Enables additional `write_stream` command options. *Data_type*: string, optional

Examples

- The following command presents how to use `opt_pattern` to find the hotspots:
  ```
  opt_pattern -find -deck deck.cpaHook.xml -map_file write_stream.map -write_stream_options { -outputMacros } -dir Find_hook -layers "M2"
  ```
- The following command presents how to fix the patterns with the `hinsts` created:
  ```
  opt_pattern -find -deck deck.cpaHook.xml -map_file write_stream.map -write_stream_options { -outputMacros } -dir Find_hook -fix -layers "M2" -fix
  ```
  The following figure depicts an example of a fixed location:
  
- The following command applies double via patterns:
  ```
  opt_pattern -find -deck deck.cpaDoubleVia.xml -dir Find_via -map_file write_stream.map -write_stream_options { -outputMacros } -layers "M2"
  ```
- The following command finds the double via patterns and fixes 200 of those:
  ```
  opt_pattern -find -deck deck.cpaDoubleVia.xml -dir Find_via -map_file write_stream.map -write_stream_options { -outputMacros } -layers "M2" -fix -findLimit 200
  ```

Related Topics

- Integration with LPA and CCP chapter in the *Innovus User Guide*
- *Litho Physical Analyzer User Guide*

read_drc_markers

```
read_drc_markers
[-help]
in_file
[-incremental]
[-orient {r0 | r90 | r180 | r270 | mx | mx90 | my | my90}]
[-origin x y]
```

Loads the specified Design Rule Checking (DRC) violation marker file. Use this command to load DRC markers, which are saved with DB format, into Innovus design. The `read_drc_markers` command is useful in situations when you want to load DRC violation markers without having to load an entire design.

For example, if during an Engineering Change Order (ECO), you find that only 25 of the 300 reported violations are actual violations that need to be corrected immediately, you can:

1. Delete the other 275 violations.
2. Save the 25 actual violations in a marker file using the `write_drc_markers` command.
3. Clear the violations from the design window.
4. Load the marker file containing the saved violation markers using the `read_drc_markers` command.

You can also use the `read_drc_markers` command in situations where, when correcting violations, the ECO editor does not make a change in the way that you wanted. Because the editor removed the marker when it made the change, you cannot find the original violation in the design. You can use the `read_drc_markers` command to load the original marker file, then click on the violation, and zoom to the area in which you were working.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `read_drc_markers` parameter. For a detailed description of the command and all its parameters, use the `man` command: `man read_drc_markers`
in_file	Specifies the name of the marker file to load. *Data_type*: string, required
`-incremental`	Loads the markers stored in the specified file incrementally. This means that the markers already loaded in memory are not removed before loaded from the specified file. *Data_type*: bool, optional
`-orient {r0 | r90 | r180 | r270 | mx | mx90 | my | my90}`	
	Rotates all the DRCs available in the violation marker file with the specified orientation. *Data_type*: enum, optional
`-origin x y`	Specifies the origin coordinate of the DRC available in the violation marker file. *Data_type*: point, optional)

Example

The following command loads markers that were previously saved to the `VG.drc` file using `saveDrc`:

```
read_drc_markers VG.drc
```

Related Information

- The following sections in the Identifying and Viewing Violations chapter of the *Innovus User Guide*:
  - Viewing Violations With the Violation Browser
  - Clearing Violations

read_markers

```
read_markers
[-help]
in_file fileName
[-orient {r0 r90 ...}]
[-rule_map_file mapName]
-type {Assura | Calibre | PVS | Hercules | ICV | CDNLitho | CDNCMP | inShapeLitho | CalibreLitho | DRV | ClockTran | CLP | VerifyPower}
[-x_offset value]
[-y_offset value]
```

Loads a violation marker file or a hotspot interface format (HIF) file in one of the following formats and creates markers that Innovus can interpret. Use this command to load third-party markers into Innovus:

- Assura™
- Calibre
- PVS
- Hercules
- ICV
- Cadence or Calibre HIF
- Chemical and Mechanical Polishing (CMP) HIF
- inShapeLitho
- Calibre litho hotspot
- NanoRoute litho
- Transition violation report
- Low power violation reports

After you load the file, you can view the markers in the Violation Browser. By cross-probing between Violation Browser and the design area, you can analyze the violations and take steps to resolve them. The Innovus Auto Query feature displays the rule names for the markers.

Parameters

`-help`	Prints a brief description that includes the type and default information for each `read_markers` parameter. For a detailed description of the command and all its parameters, use the `man` command: `man read_markers`
`in_file` *fileName*	Specifies the marker file to load. Assura files must have a `.err` extension. Calibre files must have a `.db` extension. *Data_type*: string, required
`-orient {`*r0 r90* `...}`	Supports advanced orientation transforms for loading rotated DRC violation markers before loading. This helps in overlaying the *Data_type*: string, optional

`-rule_map_file` *mapName*	Specifies the rule map file for a PVS or Calibre Litho report. The rule map file provides the layer/error type information.
	`read_markers` saves this information to the database. When the markers are created, the layer, error type, and object information
	(width, spacing, enclosure, area, minstep, etc). The rule map file should be a txt file with 3 columns:
	Rule_name Innovus_error_type layer_name
	Below is an example:
	`M1.W_4.5 width metal1`
	`M1.A_0.048 minhole metal1`
	`M1.M_1 minstep metal1`
	Data_type: string, optional

`-type {Assura | Calibre | PVS | Hercules | ICV | CDNLitho | CDNCMP | inShapeLitho | CalibreLitho | DRV | ClockTran | CLP | VerifyPower}`	
	Specifies the file type. The following file types contain violation markers: • Assura - For Assura violations • Calibre - For Calibre violations • PVS - For PVS violations • Hercules - For Hercules violations • ICV - For ICValidator report violations • CDNCMP - For Cadence CMP Predictor (CCP) violations • DRV - For transition violations (reportTranViolation) • ClockTran, CLP, and VerifyPower - For low power violations The following file types contain hotspot markers, which are used to identify areas susceptible to lithography problems: • CDNLitho (LPA as well as NanoRoute litho HIF) • inShapeLitho • CalibreLitho *Data_type*: enum, required
`-x_offset` *value*	Specifies the X offset. When you specify the X and Y offsets, Violation Browser displays the violations at the correct coordinates *Data_type*: float, optional
`-y_offset` *value*	Specifies the Y offset, which Violation Browser takes into account to displays violations at the correct coordinates. *Data_type*: float, optional

Examples

- The following command loads a CDN Litho marker file named `LPA.hif`:
  ```
  read_markers -type CDNLitho -x_offset 0.05 -y_offset 0.05 in_file LPA.hif
  ```
- The following command loads a NanoRoute litho repair file named `NR.hif`:
  ```
  read_markers -type CDNLitho in_file NR.hif
  ```

Related Information

The Identifying and Viewing Violations chapter in the *Innovus User Guide.*

report_markers

```
report_markers
[-help]
[-all]
[-ac_limit]
[-connectivity]
[-density]
[-geometry]
[-max_error_per_type integer]
[-no_false_marker]
[-overlap]
[-process_antenna]
[-short]
[-area {x1 y1 x2 y2}]
[-report filename]
[-filter filterString]
[-filter_mode {AND | OR | NOT}]
[-search_description]
```

Reports violations flagged by the verification commands. The report file and violation browser list the violations in the same order.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `report_markers` parameter. For a detailed description of the command and all its parameters, use the `man` command: `man report_markers`
`-ac_limit`	Reports AC limit violations. *Data_type*: bool, optional
`-all`	Reports all violations. *Data_type*: bool, optional
`-area {x1 y1 x2 y2}`	Specifies the coordinates of the area for which to report violations. For example, `report_markers -area {0 0 100 100}`. *Data_type*: string, optional
`-connectivity`	Reports connectivity violations. *Data_type*: bool, optional
`-density`	Reports metal density violations. *Data_type*: bool, optional
`-filter filterString`	Specifies a text string in the violation message (in the bottom portion of the Violation Browser). When you specify a text string, the Violation Browser searches the message of each violation and displays only the violations whose messages match the search conditions. To specify a list of strings, separate each string with a space. To specify a literal string with space in it, enclose the string in double quotation marks. For example, you can specify strings such as the following: • `METAL5` • `(600, 500)` • `METAL3 (700, 400)` • `"Pin of Net"` *Data_type*: string, optional
`-filter_mode {AND | OR | NOT}`	

	Specifies the condition for searching multiple strings in the violation browser. You cannot filter complex expressions because the AND, OR, and NOT parameters are mutually exclusive. • The `AND` and `OR` conditions work on single strings or multiple strings. • The `NOT` condition works only on a single string. *Data_type*: string, optional
`-geometry`	Reports geometry violations. *Data_type*: bool, optional
`-max_error_per_type` *integer*	
	Specifies the error marker number limit for each type of error. Specifies a limit for the number of violations parsed for each violation type. For example, if you set `report_markers -max_error_per_type 500`, the Violation Browser parses and displays only 500 markers per error type in the report. Use this parameter if you are loading a large DRC file with millions of violations to prevent the database from becoming slow while loading the file and to limit the number of violations you review in the browser. By default, the Violation Browser displays **all** markers in the report. *Data_type*: int, optional
`-no_false_marker`	Prohibits Innovus from displaying false violations. *Data_type*: bool, optional
`-overlap`	Reports overlap violations. *Data_type*: bool, optional
`-process_antenna`	Reports process antenna violations. *Data_type*: bool, optional
`-report` *filename*	Specifies the report file for the violation information. *Data_type*: string, optional
`-search_description`	Searches the description field for each marker in addition to the table entries. *Data_type*: bool, optional
`-short`	Reports short violations. *Data_type*: bool, optional

Examples

The following command reports all violations whose marker title contains `METAL3`:

```
report_markers -all -filter METAL3
```

The following command reports geometry violations whose marker title contains `METAL3` or `tdsp`:

```
report_markers -geometry -filter {METAL3 tdsp} -filter_mode OR
```

The following command reports geometry violations whose marker title or marker description field contains `Min` and `0.24` and `METAL3`:

```
report_markers -geometry -filter {Min 0.24 METAL3} -filter_mode AND -search_description
```

Related Information

Identifying and Viewing Violations chapter in the *Innovus User Guide*

set_isolated_cut_rule

```
set_isolated_cut_rule
[-help]
-spacing spacing1
[-corner_spacing spacing2]
[-layers {cutLayer_list}]
```

Specifies spacing rules for isolated cut layers. Use this command before running the `check_isolated_cuts` command to set the spacing rules against which `check_isolated_cut` should check isolated cuts.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `set_isolated_cut_rule` parameter. For a detailed description of the command and all its parameters, use the `man` command: `man set_isolated_cut_rule`
`-corner_spacing` *spacing2*	Specifies the maximum distance *spacing2* for the corner check. If all corners of the cut being validated have another cut within *spacing2* in the quadrant, the cut being checked is not a violation. **Note:** `-conner_spacing` is ignored if *spacing2* is smaller than `-spacing` *spacing1*. *Data_type*: float, optional
`-layers` `{`*cutLayer_list*`}`	Sets the spacing rule for specified cut layers. If this parameter is not specified, the rule is applied for all cut layers. *Data_type*: string, optional
`-spacing` *spacing1*	Specifies the maximum distance between the cut being validated to its nearest cut. If the actual distance is less than or equal to *spacing1*, `check_isolated_cut` does not report a violation. If the actual distance between the cut being validated and its nearest cut is greater than *spacing1*, `check_isolated_cut` reports a violation. *Data_type*: float, required

Example

- The following command specifies that the maximum distance between the cut being validated and the nearest cut should be `0.5`:

  ```
  set_isolated_cut_rule -spacing 0.5
  ```

- The following set of command specifies that the maximum distance between the cut being validated and the nearest cut should be `0.1`. In addition, the maximum distance for the corner check should be `0.5`. All cuts that violate these spacing rules are marked:

  ```
  set_isolated_cut_rule -spacing 0.1 - corner_spacing 0.5
  check_isolated_cut
  ```

Related Information

The following sections in the Identifying and Viewing Violations chapter in the *Innovus User Guide* document:

- Verifying Isolated Cuts
- Viewing Violations With the Violation Browser

set_signoff_verify_design_config

```
  set_signoff_verify_design_config
[-help]
[-abort_on_layout_error {yes | no}]
[-abort_on_missing_rulecheck {yes | no}]
[-area {x1 y1 x2 y2}]
[-attach_instance_name attribute_number]
[-attach_net_name attribute_number]
[-attach_net_prop {{prop_name attr_num} ... }]
[-auto_merge_base_class string]
[-config_file file_name]
[-control_file file_name]
[-core_license license_combination]
[-die_area_as_boundary]
[-distributed n]
[-distributed_timeout seconds]
[-erc_checks]
[-error_limit n]
[-extra_options string]
[-format {stream | oasis}]
[-ignore_blockage]
[-ignore_fill]
[-keep_data key_file]
[-layer_map_file streamOut_map_file]
[-layers layer_names | -layer_range {top_layer | bottom_layer}]
[-layout_path string]
[-lib_name library name]
[-license_distributed_continue]
[-license_stacking {larger_license | same_license}]
[-license_timeout number_of_seconds]
[-merge string]
[-merge_path string]
[-mode {ALL FILLONLY NOFILL NOINSTANCES}]
[-multi_cpu n]
[-multi_thread n]
[-net_map_file string]
[-nets string]
[-no_structure_name]
[-offset {x y}]
[-output_macros]
[-process_node foundary_node]
[-pvs_fill]
[-report_file]
[-reset {parameter_name | all}]
[-rule_file_drc file_name]
[-rule_file_smart_verify_lvs file_name]
[-run_name directory_name]
[-selected_inst]
[-skip_auto_load_results]
[-stripes number]
[-structure_name structure_name]
[-transform_to_inst_master]
[-ui_data]
[-uniquify_cell_names]
[-units {100 | 200 | 400 | 800 | 1000 | 2000 | 4000 | 8000 | 10000 | 20000}]
[-working_dir working_directory]
```

Sets the global parameters for the `signoff_verify_design` command. Parameters specified with the `set_signoff_verify_design_config` command are used automatically whenever `signoff_verify_design` is being run. The settings are saved in the design database and will be restored after reloading a database saved earlier.

Use the `get_signoff_verify_design_config` command to return the current values of the `set_signoff_verify_design_config` parameters.

Parameters

`-help`	Prints a brief description of the `set_signoff_verify_design_config` command usage. For a detailed description of the command, use the `man` command: `man set_signoff_verify_design_config`
`-abort_on_layout_error {yes | no}`	Optional keyword to specify whether to continue processing of the design in case of layout error or exit. *Data_type*: enum, optional
`-abort_on_missing_rulecheck {yes | no}`	Optional keyword to specify whether to continue processing of the design in case of missed rule name. *Data_type*: enum, optional
`-area {x1 y1 x2 y2}`	Optional keyword to specify a set of coordinates within which to trim metal fill. *Data_type*: rect, optional
`-attach_instance_name attribute_number`	Optional keyword to specify the instance name to attach. *Data_type*: int, optional
`-attach_net_name attribute_number`	Optional keyword to specify the net name to attach. *Data_type*: int, optional
`-attach_net_prop {{prop_name attr_num} ... }`	Optional keyword to specify the net property to attach. *Data_type*: string, optional
`-auto_merge_base_class string`	Optional keyword to specify the list of cell base classes to auto merge. It must be one or combination of block, core, corner, cover, none, and pad. *Data_type*: string, optional
`-config_file file_name`	Optional keyword to specify the Pegasus configuration file. *Data_type*: string, optional
`-control_file file_name`	Optional keyword to specify PVL control file. *Data_type*: string, optional
`-core_license license_combination`	Optional keyword to set exact combination of licenses to use. *Data_type*: string, optional
`-die_area_as_boundary`	Optional keyword to write the die area as a polygon, that is, a boundary. Otherwise, the write_stream command writes the die area as a rectangle, that is, a box. *Data_type*: bool, optional
`-distributed n`	Optional keyword to run Pegasus in multi-processor mode, n is number of processors. By default, it gets data from Innovus multi CPU usage option. *Data_type*: int, optional
`-distributed_timeout seconds`	Optional keyword to specify how long to wait for the worker processes to start. *Data_type*: int, optional
`-erc_checks`	Optional keyword to enable ERC checks to run (also requires `-connect_all` option). *Data_type*: bool, optional
`-error_limit n`	Optional keyword to limit the number of output errors. *Data_type*: int, optional
`-extra_options string`	Optional keyword to specify to run Pegasus with additional set of parameters defined in the command string. *Data_type*: string, optional
`-format {stream | oasis}`	Optional keyword to specify the format of the flow. *Data_type*: enum, optional

`-ignore_blockage`	Optional keyword to ignore design blockage objects. *Data_type*: bool, optional
`-ignore_fill`	Optional keyword to ignore design fill objects. *Data_type*: bool, optional
`-keep_data key_file`	Keeps debugging data for R&D (requires a key file). *Data_type*: string, optional
`-layer_map_file streamOut_map_file`	Optional keyword to specify `write_stream` layer map file. *Data_type*: string, optional
`-layer_range {top_layer | bottom_layer}`	Optional keyword to specify top layer name and bottom layer name for selected range. *Data_type*: string, optional
`-layers layer_names`	Optional keyword to specify that only rules that use selected layers will be processed. *Data_type*: string, optional
`-layout_path string`	Optional keyword to specify path to layout file to use for the run instead of streaming out. *Data_type*: string, optional
`-lib_name library name`	Optional keyword to specify library name. *Data_type*: string, optional
`-license_distributed_continue`	Optional keyword to specify that if required number of licenses cannot be obtained, continue with fewer CPUs based on the number of licenses obtained. *Data_type*: bool, optional
`-license_stacking {larger_license | same_license}`	Optional keyword to specify license borrowing behavior. *Data_type*: enum, optional
`-license_timeout number_of_seconds`	Optional keyword to specify how long to wait (in seconds) if a license is not immediately available when the run starts. The default is 300 seconds. *Data_type*: int, optional
`-light`	Enables rule selection preset for the light check. *Data_type*: bool, optional
`-merge string`	Optional keyword to specify the list of external stream files. *Data_type*: string, optional
`-merge_path string`	Optional keyword to specify the list of directories with external stream files. *Data_type*: string, optional
`-mode {ALL | FILLONLY | NOFILL | NOINSTANCES}`	Optional keyword to specify the mode. It identifies the layers to write. *Data_type*: enum, optional
`-multi_cpu n`	Optional keyword to specify number of workers. *Data_type*: int, optional
`-multi_thread n`	Optional keyword to specify number of threads per worker. The default is 1. *Data_type*: int, optional
`-net_map_file string`	Optional keyword to specify the path to file containing net name mapping information. *Data_type*: string, optional

`-nets string`	Specifies the list of net names to check in the `SmartVerify` LVS runs. *Data_type*: string, optional
`-no_structure_name`	Optional keyword to merge GDSII file without loading an Innovus database when the `-merge` parameter is also specified. *Data_type*: bool, optional
`-offset {x y}`	Optional keyword to specify, in microns, the location of the design in the GDSII output file. *Data_type*: point, optional
`-output_macros`	Optional keyword to write LEF abstract information such as LEF pin geometries and obstructions for macros. *Data_type*: bool, optional
`-process_node foundary_node`	Optional keyword to specify process node of used rule deck. *Data_type*: string, optional
`-pvs_fill`	Optional keyword to merge the PVS fill data attached to the current database into the output GDSII/OASIS. *Data_type*: bool, optional
`-report_file`	Optional keyword to output cell name changed to the file. *Data_type*: bool, optional
`-rule_file_drc file_name`	Optional keyword to specify rule file for DRC run. *Data_type*: string, optional
`-rule_file_smart_verify_lvs file_name`	Optional keyword to specify rule file for the `smart_verify_lvs` run. *Data_type*: string, optional
`-run_name directory_name`	Optional keyword to specify name of run directory. *Data_type*: string, optional
`-selected_inst`	Runs only the selected instance. *Data_type*: bool, optional
`-skip_auto_load_results`	Optional keyword to specify not to run Pegasus RV after successful run. *Data_type*: bool, optional
`-stripes number`	Optional keyword to specify the number of stripes that divide the design. The default is 1. *Data_type*: int, optional
`-structure_name structure_name`	Optional keyword to specify the superstructure that consists of all geometries that the map file writes out. *Data_type*: string, optional
`-transform_to_inst_master`	Transforms data to the selected instances master coordinate space. *Data_type*: bool, optional
`-ui_data`	Optional keyword to save data required for LVS/`smart_verify_lvs` probing and debugging. *Data_type*: bool, optional
`-uniquify_cell_names`	Optional keyword to create unique cell names if there are name collisions when the software merges files. *Data_type*: bool, optional
`-units {100 | 200 | 400 | 800 | 1000 | 2000 | 4000 | 8000 | 10000 | 20000}`	

	Optional keyword to specify the resolution for values in the GDSII/OASIS file. *Data_type*: enum, optional
`-working_dir` *working_directory*	Optional keyword to specify working directory. *Data_type*: string, optional

Examples

Following are few examples of the set_signoff_verify_design_config command for defining other options required for the run:

- ```
  set_signoff_verify_design_config \
  -output_macros -die_area_as_boundary \
  -rule_file_drc $drcDeck\
  -layer_map_file $gdsOutMap \
  -control_file $drcCtrl \
  -units 2000 \
  -multi_cpu 4 -multi_thread 4
  source merge.txt
  ```
- ```
  set_signoff_verify_design_config \
  -format stream \
  -merge $merge
  ```
- ```
  set_signoff_verify_design_config
  -process_node EXAMPLE_NODE \
  -process_node_data "./RULE/my_custom_template"
  ```

Related Information

- get_signoff_verify_design_config
- signoff_verify_design
  ```

set_snap_grid

```
set_snap_grid
[-help]
-layers {layer_list}
[-pitch x y]
```

Creates a global snapping grid for wiring (including metal fill shapes and via cuts). The grid defines the snap pitch, which is finer than the LEF `PITCH` for the specified layer. The grid is a multiple of the manufacturing grid. You can specify pitch value to check/set routing/metal fill segment's corner edges on the snap grid.

When you specify this command, the wires, metal fill shapes, and via cuts are snapped to this grid, and the edges of the shapes (for example, wires and vias) lie on the manufacturing grid.

The snap grid is not saved in the Innovus database. To use the snap grid, you must run this command each time you load a design.

Verify geometry, power planning and routing, metal fill, and wire editing share the grid setting. Before running commands related to these areas, use `set_snap_grid` to define the grid as additional constraints.

For more information, see the commands `add_metal_fill`, `trim_metal_fill`, and `add_via_fill`.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `set_snap_grid` parameter. For a detailed description of the command and all its parameters, use the `man` command: `man set_snap_grid`
`-layers {layer_list}`	Specifies the layers for the snap grid. You can specify metal layers as METAL1, METAL2, and so on, and cut layers as VIA12, VIA23, and so on. *Data_type*: string, optional
`-pitch x y`	Specifies the pitch for the snap grid. The LEF `PITCH` must be a multiple (for example, 2x, 4x, 8x) of the snap grid for the specified layer. For example, if the LEF `PITCH` is 0.2 µ, you can specify a snap grid pitch for both x and y of `0.1` µ. *Data_type*: point, optional

Related Information

- The Viewing Violations With the Violation Browser section in the Identifying and Viewing Violations chapter in the *Innovus User Guide* document
- Optimizing Metal Density chapter in the *Innovus User Guide*

signoff_verify_design

```
  signoff_verify_design
[-help]
[-abort_on_layout_error {yes | no}]
[-abort_on_missing_rulecheck {yes | no}]
[-area {x1 y1 x2 y2}]
[-attach_instance_name attribute_number]
[-attach_net_name attribute_number]
[-attach_net_prop {{prop_name attr_num} ... }]
[-auto_merge_base_class string]
[-beol string | -feol string | -light]
[-background_mode]
[-config_file file_name]
[-control_file file_name]
[-core_license license_combination]
[-die_area_as_boundary]
[-distributed n]
[-distributed_timeout seconds]
[-drc | -merge_library | -generate_context | -smartconnect | -smart_verify_lvs | -smart_verify_drc | -load_results path_to_rundir]
[-erc_checks]
[-error_limit n]
[-extra_options string]
[-format {stream | oasis}]
[-ignore_blockage]
[-ignore_fill]
[-layer_map_file streamOut_map_file]
[-layers layer_names | -layer_range {top_layer | bottom_layer}]
[-layout_path string]
[-lib_name library name]
[-license_distributed_continue]
[-license_stacking {larger_license | same_license}]
[-license_timeout number_of_seconds]
[-lsf]
[-merge string]
[-merge_path string]
[-mode {ALL FILLONLY NOFILL NOINSTANCES}]
[-multi_cpu n]
[-multi_thread n]
[-net_map_file string]
[-nets string]
[-no_structure_name]
[-offset {x y}]
[-output_macros]
[-process_node foundary_node]
[-pvs_fill]
[-report_file]
[-rule_file_drc file_name]
[-rule_file_smart_verify_lvs file_name]
[-run_name directory_name]
[-selected_inst]
[-skip_auto_load_results]
[-stripes number]
[-structure_name structure_name]
[-transform_to_inst_master]
[-ui_data]
[-uniquify_cell_names]
[-units {100 | 200 | 400 | 800 | 1000 | 2000 | 4000 | 8000 | 10000 | 20000}]
[-working_dir working_directory]
```

Enables configurable signoff checking at every design stage. It integrates Pegasus Signoff and Pegasus SmartVerify™ with Innovus. These checks help you maintain your designs DRC and LVS clean through the design cycle, and help achieve zero DRC and zero shorts out of Innovus. Running signoff checks late in the design involves long iterative runtimes, debug, fixing, and validation cycles. This impacts time to tapeout. This command allows you to run the signoff checks simultaneously without interrupting Innovus.

Parameters

`-help`	Prints a brief description of the `signoff_verify_design` command usage. For a detailed description of the command, use the `man` command: `man signoff_verify_design`
`-abort_on_layout_error {yes | no}`	Optional keyword to specify whether to continue processing of the design in case of layout error or exit. *Data_type*: enum, optional
`-abort_on_missing_rulecheck {yes | no}`	Optional keyword to specify whether to continue processing of the design in case of missed rule name. *Data_type*: enum, optional
`-area {x1 y1 x2 y2}`	Optional keyword to specify a set of coordinates within which to trim metal fill. *Data_type*: rect, optional
`-attach_instance_name attribute_number`	Optional keyword to specify the instance name to attach. *Data_type*: int, optional
`-attach_net_name attribute_number`	Optional keyword to specify the net name to attach. *Data_type*: int, optional
`-attach_net_prop {{prop_name attr_num} ... }`	Optional keyword to specify the net property to attach. *Data_type*: string, optional
`-auto_merge_base_class string`	Optional keyword to specify the list of cell base classes to auto merge. It must be one or combination of block, core, corner, cover, none, and pad. *Data_type*: string, optional
`-background_mode`	Optional keyword to run SVD job in background mode. *Data_type*: bool, optional
`-beol string`	Optional keyword to provide list of rule selection blocks for BEOL checks. *Data_type*: string, optional
`-config_file file_name`	Optional keyword to specify the Pegasus configuration file. *Data_type*: string, optional
`-control_file file_name`	Optional keyword to specify PVL control file. *Data_type*: string, optional
`-core_license license_combination`	Optional keyword to set exact combination of licenses to use. *Data_type*: string, optional
`-die_area_as_boundary`	Optional keyword to write the die area as a polygon, that is, a boundary. Otherwise, the `write_stream` command writes the die area as a rectangle, that is, a box. *Data_type*: bool, optional
`-distributed n`	Optional keyword to run Pegasus in multi-processor mode, n is number of processors. By default, it gets data from Innovus multi CPU usage option. *Data_type*: int, optional
`-distributed_timeout seconds`	Optional keyword to specify how long to wait for the worker processes to start. *Data_type*: int, optional
`-drc`	Required keyword to run DRC check. *Data_type*: bool, required
`-erc_checks`	Optional keyword to enable ERC checks to run (also requires `-connect_all` option). *Data_type*: bool, optional
`-error_limit n`	Optional keyword to limit the number of output errors. *Data_type*: int, optional

`-extra_options` *string*	Optional keyword to specify to run Pegasus with additional set of parameters defined in the command string. *Data_type*: string, optional
`-feol` *string*	Optional keyword to provide list of rule selection blocks for FEOL checks. *Data_type*: string, optional
`-format {stream | oasis}`	Optional keyword to specify the format of the flow. *Data_type*: enum, optional
`-generate_context`	Generates context for an instance. *Data_type*: bool, required
`-ignore_blockage`	Optional keyword to ignore design blockage objects. *Data_type*: bool, optional
`-ignore_fill`	Optional keyword to ignore design fill objects. *Data_type*: bool, optional
`-layer_map_file` *streamOut_map_file*	Optional keyword to specify `write_stream` layer map file. *Data_type*: string, optional
`-layer_range {top_layer | bottom_layer}`	Optional keyword to specify top layer name and bottom layer name for selected range. *Data_type*: string, optional
`-layers` *layer_names*	Optional keyword to specify that only rules that use selected layers will be processed. *Data_type*: string, optional
`-layout_path` *string*	Optional keyword to specify path to layout file to use for the run instead of streaming out. *Data_type*: string, optional
`-lib_name` *library name*	Optional keyword to specify library name. *Data_type*: string, optional
`-license_distributed_continue`	Optional keyword to specify that if required number of licenses cannot be obtained, continue with fewer CPUs based on the number of licenses obtained. *Data_type*: bool, optional
`-license_stacking {larger_license | same_license}`	Optional keyword to specify license borrowing behavior. *Data_type*: enum, optional
`-license_timeout` *number_of_seconds*	Optional keyword to specify how long to wait (in seconds) if a license is not immediately available when the run starts. The default is 300 seconds. *Data_type*: int, optional
`-light`	Enables rule selection preset for the light check. *Data_type*: bool, optional
`-load_results` *path_to_rundir*	Required keyword to run Pegasus RV with data from provided run directory. *Data_type*: string, optional
`-lsf`	Optional keyword to specify to distribute the run to different machines using LSF. *Data_type*: bool, optional
`-merge` *string*	Optional keyword to specify the list of external stream files. *Data_type*: string, optional
`-merge_library`	Generates single merged layout from the list of layouts to merge. *Data_type*: bool, required

`-merge_path` *string*	Optional keyword to specify the list of directories with external stream files. *Data_type*: string, optional
`-mode {ALL | FILLONLY | NOFILL | NOINSTANCES}`	Optional keyword to specify the mode. It identifies the layers to write. *Data_type*: enum, optional
`-multi_cpu` *n*	Optional keyword to specify number of workers. *Data_type*: int, optional
`-multi_thread` *n*	Optional keyword to specify number of threads per worker. The default is 1. *Data_type*: int, optional
`-net_map_file` *string*	Optional keyword to specify the path to file containing net name mapping information. *Data_type*: string, optional
`-nets` *string*	Specifies the list of net names to check in the `SmartVerify` LVS runs. *Data_type*: string, optional
`-no_structure_name`	Optional keyword to merge GDSII file without loading an Innovus database when the `-merge` parameter is also specified. *Data_type*: bool, optional
`-offset {`*x y*`}`	Optional keyword to specify, in microns, the location of the design in the GDSII output file. *Data_type*: point, optional
`-output_macros`	Optional keyword to write LEF abstract information such as LEF pin geometries and obstructions for macros. *Data_type*: bool, optional
`-process_node` *foundary_node*	Optional keyword to specify process node of used rule deck. *Data_type*: string, optional
`-pvs_fill`	Optional keyword to merge the PVS fill data attached to the current database into the output GDSII/OASIS. *Data_type*: bool, optional
`-report_file`	Optional keyword to output cell name changed to the file. *Data_type*: bool, optional
`-rule_file_drc` *file_name*	Optional keyword to specify rule file for DRC run. *Data_type*: string, optional
`-rule_file_smart_verify_lvs` *file_name*	Optional keyword to specify rule file for the `smart_verify_lvs` run. *Data_type*: string, optional
`-run_name` *directory_name*	Optional keyword to specify name of run directory. *Data_type*: string, optional
`-selected_inst`	Runs only the selected instance. *Data_type*: bool, optional
`-skip_auto_load_results`	Optional keyword to specify not to run Pegasus RV after successful run. *Data_type*: bool, optional
`-smartconnect`	Runs `smartconnect` check. *Data_type*: bool, optional
`-smart_verify_drc`	Required option to run `smart_verify_drc` check. *Data_type*: bool, optional
`-smart_verify_lvs`	Required option to run `smart_verify_lvs` check. *Data_type*: bool, optional

`-stripes` *number*	Optional keyword to specify the number of stripes that divide the design. The default is 1. *Data_type*: int, optional
`-structure_name` *structure_name*	Optional keyword to specify the superstructure that consists of all geometries that the map file writes out. *Data_type*: string, optional
`-transform_to_inst_master`	Transforms data to the selected instances master coordinate space. *Data_type*: bool, optional
`-ui_data`	Optional keyword to save data required for LVS/`smart_verify_lvs` probing and debugging. *Data_type*: bool, optional
`-uniquify_cell_names`	Optional keyword to create unique cell names if there are name collisions when the software merges files. *Data_type*: bool, optional
`-units {100 | 200 | 400 | 800 | 1000 | 2000 | 4000 | 8000 | 10000 | 20000}`	
	Optional keyword to specify the resolution for values in the GDSII/OASIS file. *Data_type*: enum, optional
`-working_dir` *working_directory*	Optional keyword to specify working directory. *Data_type*: string, optional

Examples

- The following command prevents auto-loading of results in Results Viewer, after completion of the software run:
  ```
  signoff_verify_design -drc -working_dir DRC_rundir3 -run_name DRCrun -skip_auto_load_results
  ```
  This will not open the Results Viewer with results once the run completes. To load results from the run, use the `-load_results` parameter, as shown in the example usage below:
  ```
  signoff_verify_design -load_results DRC_rundir3/drc/DRCrun
  ```
- The following command runs DRC and cross probes the run results using the Design Review:
  ```
  signoff_verify_design -drc -working_dir DRC_rundir4 -run_name DRCrun
  ```
- The following command runs DRC checks on a specified area:
  ```
  signoff_verify_design -drc -area {30 0 45 15} -working_dir DRC_rundir5 -run_name beolArea
  ```
- The following command runs DRC checks on a specified layers:
  ```
  signoff_verify_design -drc -layer_range {Metal4 Metal5} -working_dir DRC_rundir6 -run_name beolM4M5
  ```
- The following command specifies a single layer to run checks:
  ```
  signoff_verify_design -drc -layers {Metal4 Metal4} -working_dir DRC_rundir7 -run_name beolM4
  ```
- Following is an example of using the `-beol`, `-layer_range`, `-working_dir`, and `-run_name` parameters together:
  ```
  signoff_verify_design -drc -beol {all} -layer_range {Metal2 Metal4} -working_dir workdir_drc -run_name beol_M2M4
  ```
- Following is an example of using the `-beol`, `-area`, `-working_dir`, and `-run_name` parameters together:
  ```
  signoff_verify_design -drc -beol {all} -area {30 0 45 15} -working_dir workdir_drc -run_name beolArea
  ```

Related Information

- `get_signoff_verify_design_config`
- `set_signoff_verify_design_config`

write_drc_markers

```
write_drc_markers
[-help]
out_file
[-force]
[-selected]
```

Saves DRC violation markers in the specified file. The `write_drc_markers` command is useful for situations where you want to save DRC violation markers without having to save the entire design.

For example, if during ECO you find that only 25 of the 500 reported violations are actual violations that need to be corrected immediately, you can:

1. Delete the other 475 violations.
2. Save the 25 violations in a marker file using the `write_drc_markers` command.
3. Clear the violations from the design window.
4. Load the marker file containing the saved violation markers using the `read_drc_markers` command.

Use the `write_drc_markers` command after markers have been flagged by any of the following commands:

- `check_metal_density`
- `check_process_antenna`

Parameters

`-help`	Prints a brief description that includes the type and default information for each `write_drc_markers` parameter. For a detailed description of the command and all its parameters, use the `man` command: `man write_drc_markers`
`out_file`	Saves the DRC violation markers in a file with the specified name. *Data_type*: string, required
`-force`	Creates drc file with a 0 drc count. **Note:** When there is no drc in the design, drc file is not generated (Default). *Data_type*: bool, optional
`-selected`	Saves only the selected DRC violation markers. *Data_type*: bool, optional

Examples

The following command saves markers to the `VG.drc` file:

```
write_drc_markers VG.drc
```

Related Topics

- `check_drc`
- The following sections in the Identifying and Viewing Violations chapter of the *Innovus User Guide* document:
  - Viewing Violations With the Violation Browser
  - Clearing Violations

6

Clock Tree Synthesis Commands and Attributes

- add_clock_tree_exclusion_drivers
- add_clock_tree_source_group_roots
- assign_clock_tree_source_groups
- ccopt_design
- ccopt Category Attributes
- check_clock_tree_convergence
- check_cts_config
- clock_design
- clock_opt_design
- clock_post_route_repair
- commit_clock_tree_route_attributes
- connect_clock_tree_mesh_drivers
- convert_lib_clock_tree_latencies
- create_clock_tree
- create_clock_tree_source_group
- create_clock_tree_spec
- create_flexible_htree
- create_generated_clock_tree
- create_route_type
- create_skew_group
- cts Category Attributes
- delete_clock_tree_drivers
- delete_clock_tree_source_group
- delete_clock_tree_spec
- delete_clock_trees
- delete_flexible_htrees
- delete_skew_groups
- get_ccopt_property_common_ui_map
- get_skew_group_path
- get_clock_tree_capacitance
- get_clock_tree_effective_max_capacitance
- get_clock_tree_transition_time
- get_cts_graph_traversal
- get_lib_clock_tree_path_delay
- get_skew_group_delay
- gui_ctd_close
- gui_ctd_get_title
- gui_ctd_get_window_id
- gui_ctd_highlight

- gui_ctd_open
- gui_ctd_write_histogram
- gui_ctd_write_view
- gui_ctd_set_title
- gui_ctd_zoom
- merge_clock_tree_instances
- place_detail_clock_tree
- read_cts_config
- report_cts_cell_name_info
- report_ccopt_worst_chain
- report_clock_tree_convergence
- report_clock_tree_halo_violations
- report_clock_tree_structure
- report_clock_trees
- report_cts_cell_filtering_reasons
- report_pin_insertion_delays
- report_skew_groups
- reset_cts_config
- reset_clock_tree_routing_state
- route_flexible_htrees
- route_clock_tree_nets
- synthesize_flexible_htrees
- update_clock_tree_source_latency
- update_clock_tree_spec_annotations
- update_skew_group
- write_cts_config

add_clock_tree_exclusion_drivers

```
add_clock_tree_exclusion_drivers
[-help]
[-base_cell list_library_cells]
```

Removes the excluded sinks from the clock trees by adding exclusion drivers above these sinks. Use this command after running the `create_clock_tree_spec` command. Alternatively, you can use the `cts_add_exclusion_drivers` attribute to remove the exclude sinks from the clock tree. It is recommended that you use one of the two, either the command or the attribute, to exclude sinks from the clock trees.

The names of the newly added drivers start with '`AZ_cse`', so that they can be easily identified. This functionality is useful because it allows you to fix hold violations on clock nets. GigaOpt does not fix hold violations on clock nets. For example, if a pin, D, of a flop is a clock sink then the net attached to the D pin is a clock net. Any hold violations will not be fixed by GigaOpt. However, by adding an exclusion buffer above the D pin, the net between the buffer's output pin and the D pin of the flop becomes a non-clock net and any hold violations will be fixed by GigaOpt. Therefore, this new functionality allows previously unfixable hold violations to be fixed.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `add_clock_tree_exclusion_drivers` parameter. For a detailed description of the command and all of its parameters, use the man command: `man add_clock_tree_exclusion_drivers`.
`-base_cell`	Specifies a particular buffer or inverter to use as an exclusion driver. If this parameter is not specified or if the attribute, `cts_add_exclusion_drivers`, is used to enable this functionality, then CCOpt will choose small area drivers from the CTS cell lists to use as exclusion drivers. CCOpt will either add pairs of inverters, or single buffers as exclusion drivers. While choosing the cell to use, CCOpt will choose either inverters or buffers depending on the value specified using the `cts_use_inverters` attribute. This attribute specifies whether clock tree synthesis should use inverters or buffers when balancing the clock trees. *Data_type*: string, optional

Related Information

- `create_clock_tree_spec`
- The following attributes:
  - `cts_add_exclusion_drivers`
  - `cts_effective_sink_type`
  - `cts_implicit_sink_type`
  - `cts_sink_type`
  - `cts_use_inverters`

  For details of the above attributes, see the cts Category Attributes chapter.
- The following section in Clock Tree Synthesis chapter of *Innovus Stylus Common UI User Guide*
  - Shared Clock and Data Concerns

add_clock_tree_source_group_roots

```
add_clock_tree_source_group_roots
[-help]
-cell driverCell
[-locations {{x1 y1} {x2 y2}...}]
-name sourceGroupName
-pin rootPin
[-grid {columns rows} [[-adjust_grid_for_aspect_ratio] [-grid_rect {lx ly ux uy}]
[-grid_search_area {width height}]
[-grid_exclusion_rects {{lx1 ly1 ux1 uy1} {lx2 ly2 ux2 uy2}...}]]]
```

This command updates the design and clock tree specification to create a multiple-tree "source group" below the specified driving root pin. A source group is a set of multiple, logically equivalent, clock trees that drive a collection of clock tree sinks. The actual allocation of sinks to specific source group roots is chosen during clock tree synthesis.

This command performs the following tasks:

- Inserts the instances that will become the clock tree roots into the netlist. Their input pins will be driven by the root pin and the fanout originally driven by the root pin will be rewired to the output of a single root instance.
- Defines clock trees at the outputs of each of the newly inserted root instances.
- Defines a clock source group to combine the clock trees.

You can insert a grid of tap buffers and also omit a tap buffer at grid points that are blocked. For this, you can specify grid coordinates, a two-value pair `{x y}`, that place the source group driver at that location.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `add_clock_tree_source_group_roots` parameter. For a detailed description of the command and all of its parameters, use the man command: `man add_clock_tree_source_group_roots`.
`-adjust_grid_for_aspect_ratio`	
When specified, exchanges rows and columns to match the aspect ratio. *Data_type*: bool, optional	
`-cell driverCell`	
	Specifies the driver cell to use for new source group roots (`base_cell`). This is a required parameter. *Data_type*: (base_cell), required
`-grid {columns rows}`	
	Specifies a grid of target locations. By default, the command attempts to create a full grid consisting of columns and rows. However, if a particular grid target point falls within an exclusion rectangle, specified using the `-grid_exclusion_rects` parameter, then that grid point will be skipped. If the `-adjust_grid_for_aspect_ratio` parameter is specified, the columns and rows will be swapped if doing so better matches the aspect ratio of the grid rectangle area. **Note**: The `-grid` and `-locations` parameters can be used together. For example, you can specify a grid, and then use `-locations` to add a few additional taps in specific floor-plan locations. *Data_type*: integer, optional
`-grid_exclusion_rects {{lx1 ly1 ux1 uy1} {lx2 ly2 ux2 uy2}...}`	
	Provides the coordinates of rectangular areas that should not contain source group roots. *Data_type*: (rect)+, optional
`-grid_rect {lx ly ux uy}`	

	Specifies the area that the grid should cover. This parameter provides the overall rectangle intended to be driven by the taps requested by the `-grid` parameter. **Note**: This parameter does not specify the bounding box of the grid taps. Basically the overall `-grid_rect` rectangle is cut up into `m x n`smaller rectangles, and a tap is targeted at the center of each smaller rectangle. to understand more clearly, consider the below example. If you have `-grid {2 2}` and `-grid_rect {0 0 100 100}`, the corresponding target locations will be `{{25 25} {25 75} {75 25} {75 75}}`. This is different than if the `-grid_rect` were the bounding box of tap locations, in which case it would be `{{0 0} {0 100} {100 0} {100 100}})`. *Data_type*: rect, optional
`-grid_search_area {width height}`	
	Specifies the size of the search area around each grid point. *Data_type*: point, optional
`-locations {{x1 y1} {x2 y2}...}`	
	Specifies a list of (`x,y`) locations for the new roots that are to be created. The specified roots are inserted at the nearest legal location. **Note**: This parameter takes a list of points, not rectangles. A specified point will be taken as a tap target location, and the software will search for the nearest legal location. If the `-grid_search_area {width height}` parameter is specified, the search will be limited to the points specified using the `-grid` parameter and not points specified using the `-locations` parameter. If it is not specified, the software will search as far as needed to find a legal location for a tap. If a legal location cannot be found, either within the specified search area, or anywhere in the floorplan, the tap will be added at the target location and not legalized. If all taps can be inserted at legal locations, they will be set to locked for CTS placement status. If one or more taps cannot be legalized, the command will print a warning and all taps will be left in simple placed status. **Note**: The behavior when a tap cannot be legalized, and for whether taps are locked applies to all taps created by this command - from the `-grid` and/or `-locations` parameters. *Data_type*: (point)+, optional
`-name` *prefix*	Specifies the name of the source group to be created for the roots. The name should be unique. The multi-tap roots will be named using this string as a prefix. For example, `sourceGroupName_0`, `sourceGroupName_1`..`sourceGroupName_n`. **Note**: • The skew group name will be `<prefix>_skew_group`. • The clock tree name for a tap will be `<prefix>_<index>`, and `<prefix>_<index>_inst` for the corresponding cell instance. This is a required parameter. *Data_type*: string, required
`-pin` *rootPin*	Specifies the name of the root pin that will drive the multi-tap root inputs. The pin must be an output pin and must drive a net. The entire clock structure, including clock gates, flops, and so on that are to be driven by the multi-tap roots should originally be connected below this pin. If there are multiple drivers on the root net, any one of their output pins may be specified as the `-pin` parameter. *Data_type*: (pin|port), required

Related Information

- `assign_clock_tree_source_groups`
- `create_clock_tree_source_group`
- `delete_clock_tree_source_group`
- Clock Tree Synthesis chapter of *Innovus Stylus Common UI User Guide*

assign_clock_tree_source_groups

```
assign_clock_tree_source_groups
[-help]
[-no_merge]
```

When used on designs with clock tree source group definitions, this command assigns clock sinks to the source group roots - multi-tap assignment - and then performs the cloning and rewiring necessary to distribute the sinks among the source group roots.

Note that the source group assignment process normally happens automatically during the early stages of clock tree synthesis. However, on large designs, the full clock tree synthesis step can take some time. The purpose of this command is to provide an easier way to get early feedback about the multi-tap assignment process, visualize the result, adjust source group definitions, and so on.

By default, the command will do initial clock gate merging and then clone as necessary to implement the assignment. However, the merging step can be disabled by specifying the `-no_merge` option. If no source groups are defined, the command will do nothing.

After running the command, you can review the log file messages, or check the result in the database either by using the Tcl commands, `get_db` command, clock tree debugger, and so on.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `assign_clock_tree_source_groups` parameter. For a detailed description of the command and all of its parameters, use the man command: `man assign_clock_tree_source_groups`.
`-no_merge`	Specifies that clock gates should not be merged. *Data_type*: bool, optional

Related Information

- `add_clock_tree_source_group_roots`
- `create_clock_tree_source_group`
- `delete_clock_tree_source_group`

ccopt_design

```
ccopt_design
[-help]
[-check_cts_config]
[-expanded_views]
[-num_paths number_of_paths]
[-report_dir out_dir]
[-report_prefix file_name_prefix]
[-timing_debug_report]
```

Performs clock concurrent optimization (CCOpt) on the current loaded design in Innovus. CCOpt optimizes both the clock tree and the datapath to meet global timing constraints.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `ccopt_design` parameter. For a detailed description of the command and all of its parameters, use the man command: `man ccopt_design`.
`-check_cts_config`	Checks that all the prerequisites for running clock tree synthesis are fulfilled without actually doing CTS. For example, it checks that the design is loaded, and the options set by that the user are correct. *Data_type*: bool, optional
`-expanded_views`	This parameter is not interpreted in CCOpt but passed directly to `opt_design`. The parameter is same as `opt_design -expanded_views` parameter, which is specified to generate detailed view-specific timing reports at the end of the `opt_design` command run. *Data_type*: bool, optional
`-num_paths` *number_of_paths*	Specifies the number of paths generated in the timing reports. By default, all timing paths are reported. *Data_type*: integer, optional
`-report_dir` *out_dir*	This parameter specifies the directory where the software writes timing reports that are generated after timing optimization. The default directory is `./timingReports` *Data_type*: string, optional
`-report_prefix` *file_name_prefix*	Specifies a prefix for `opt_design` report file names. *Default*: `DesignName_DesignStage` , where `DesignStage` is pre-CTS, post-CTS, or post-Route *Data_type*: string, optional
`-timing_debug_report`	When specified, generates a machine-readable timing debug report after completing CTS. This report is used for debugging timing results using the timing debug feature in Stylus. **Note**: You can specify the number of paths to be reported using the `-num_paths` parameter. *Data_type*: bool, optional

Examples

- The following command runs a complete CCOpt optimization:
  ```
  ccopt_design
  ```

Related Information

- `create_clock_tree_spec`
- `opt_design`
- The following topic in the *Innovus Stylus Common UI User Guide*.
  - Flow and Quick Start for CTS

ccopt Category Attributes

- ccopt_auto_limit_insertion_delay_factor
- ccopt_merge_clock_gates
- ccopt_merge_clock_logic

To set the value of any root attribute, use the `set_db` command:

```
set_db attribute_name value
```

To get the current value of any root attribute, use the `get_db` command:

```
get_db attribute_name
```

To get the current value of all attributes in the `ccopt` category, use the following command:

```
get_db -category ccopt
```

Note: For attributes that are keyed, ensure that the design is loaded before using the `get_db attribute_name` command.

For more information about the object attributes, see *Stylus Common UI Database Object Information*.

ccopt_auto_limit_insertion_delay_factor

```
ccopt_auto_limit_insertion_delay_factor value
```

Default: `1.5`

Data_type: double, read/write

CCOpt attempts to keep the insertion delays of each clock tree a fixed multiple of the longest insertion delay that would result from a global skew approach. This multiple can be modified by design timing and the presence of other clock trees, but will start at a fixed fraction above the global skew insertion delay. This attribute specifies that fixed fraction.

Valid values: `real`

See also: `auto_limit_insertion_delay_factor_skew_group`

This global attribute does not use additional arguments.

ccopt_merge_clock_gates

```
ccopt_merge_clock_gates {true | false}
```

Default: true

Data_type: bool, read/write

If set to `true`, clock gate merging is enabled. If this is set to `0`, merging of all clock gates is disabled, including clock gates that may have been cloned by CTS.

See also attribute `cts_merge_clock_gates`.

Valid values: true false

ccopt_merge_clock_logic

```
ccopt_merge_clock_logic {true | false}
```

Default: true

Data_type: bool, read/write

If set to true, clock logic merging is enabled. If this is `false`, merging of all clock logics is disabled, including clock logics which may have been cloned by CTS.

See also attribute `cts_merge_clock_logic`.

Valid values: `true false`

check_clock_tree_convergence

```
check_clock_tree_convergence
[-help]
[-path_threshold threshold_value]
```

This command is used to trigger a warning message for sinks that have more than the specified number of convergence paths from the clock sources to the sink. This check is performed at the end of clock tree specification, for complex convergence above sinks that might cause large runtimes in CCOpt as it attempts to balance these trees.

When this command is specified, any sinks that are found to have more than the specified *threshold_value* of paths from the clock sources to the sink are reported in a warning message from the software. For example:

`WARN: (ENCCCOPT-4338):` There are 17 clock paths leading to primaryout. This may cause long CCOpt runtimes. Type '`man ENCCCOPT-4338`' for more detail.

The following detailed information is provided when you type `man ENCCCOPT-4338`:

```
The design contains a lot of converging clock paths leading to some sinks. This may cause long CCOpt runtimes. The sinks with the most paths
leading to them can be found with report_ccopt_clock_tree_convergence. See CCOpt Clock Tree Specification under Clock Tree Synthesis in the
User Guide for more information.
```

Note: The warning is issued if the number of clock paths to any sink is greater than the threshold value of paths specified by the command `check_clock_tree_convergence`.

To view a summary of the convergence above clock sinks and to identify the names of the sinks with the highest number of clock network paths, run the `report_clock_tree_convergence` command.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `check_clock_tree_convergence` parameter. For a detailed description of the command and all of its parameters, use the man command: `man check_clock_tree_convergence`.
`-path_threshold` *threshold_value*	Triggers a warning message for sinks with more than the specified number of paths. *Default*: 100 *Data_type*: int, optional

Related Information

- `create_clock_tree_spec`
- `report_clock_tree_convergence`
- The following section in Automatic Clock Tree Specification Creation in *Innovus Stylus Common UI User Guide*
  - Clock Tree Convergence

check_cts_config

```
  check_cts_config
[-help]
```

This command checks the current CTS configuration and displays the results in the console.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `check_cts_config` parameter. For a detailed description of the command and all of its parameters, use the man command: `man check_cts_config`.

Related Information

- `read_cts_config`
- `reset_cts_config`
- `write_cts_config`
- Clock Tree Synthesis chapter of *Innovus Stylus Common UI User Guide*

clock_design

```
clock_design
[-help]
```

Performs clock tree synthesis (CTS) on the design using the CCOpt engine. It does not perform any datapath optimization or useful skew.

Parameters

`-help`	Outputs a brief description about the `clock_design` command.

Related Information

- `create_clock_tree_spec`
- `opt_design`
- The following topics in the *Innovus Stylus Common UI User Guide*
  - Flow and Quick Start for CTS
  - CTS Configuration and Method

clock_opt_design

```
clock_opt_design
[-help]
[-check_cts_config]
[-cts]
[-enable_pre_cts_opt]
[-expanded_views]
[-num_paths number_of_paths]
[-report_dir dirname]
[-report_prefix file_name_prefix]
[-timing_debug_report]
```

The `clock_opt_design` command is used in place of `ccopt_design` when the input DB is place_opt_design V2 (PODV2). It performs clock concurrent optimization (CCOpt) on the current loaded design in Innovus. CCOpt optimizes both the clock tree and the datapath to meet global timing constraints. The only difference between this flow and the `ccopt_design` flow is that `clock_opt_design` has legalization at the beginning to remove overlapping caused by PODV2 optimization.

In this flow, the commands, `place_opt_design` and `clock_opt_design` are used in combination. This means that the `clock_opt_design` command can only be used with the `place_opt_design V2`, which is the `place_opt_design` command that is run after enabling the PODV2 flow.

This flow is enabled by setting the `opt_enable_podv2_clock_opt_flow` attribute to `true`.

```
set_db opt_enable_podv2_clock_opt_flow true

clock_opt_design
```

If you set the above attribute but the DB is not PODV2, the software gives the following message:

```
Cannot run clock_opt_design because current DB is not PODV2 DB
```

By default, this attribute is set to `false`.

The effort level for the PODV2 flow is set using the following `opt` category attribute:

```
set_db opt_podv2_flow_effort {auto | standard | extreme}
```

The following settings are available:

- `auto`: The software follows the `design_flow_effort` attribute setting to perform optimization.
- `standard`: The software performs a fast prects optimization and runtime is less. This option is recommended in the early design stage or in simple designs for PPA estimation.
- `extreme`: The software performs a deep optimization to chase the best PPA. Runtime is high.

By default, this attribute is set to `auto`.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `clock_opt_design` parameter. For a detailed description of the command and all of its parameters, use the man command: `man clock_opt_design`.
`-check_cts_config`	Checks that all the prerequisites for running clock tree synthesis are fulfilled without actually doing CTS. For example, it checks that the design is loaded, and the options set by that the user are correct. There are no changes made to the design. *Data_type*: bool, optional
`-cts`	Turns off clock concurrent optimization, and performs only clock tree synthesis (CTS) using the CCOpt engine. It does not perform any datapath optimization or useful skew. *Data_type*: bool, optional
`-enable_pre_cts_opt`	Enables pre-CTS optimization in the extreme flow. *Data_type*: bool, optional
`-expanded_views`	This parameter is not interpreted in CCOpt but passed directly to `optDesign`. The parameter is same as `opt_design -expanded_views` parameter, which is specified to generate detailed view-specific timing reports at the end of the `optDesign` command run. *Data_type*: bool, optional
`-num_paths` *number_of_paths*	Specifies the number of paths generated in the timing reports. By default, all timing paths are reported. *Data_type*: string, optional
`-report_dir` *dirname*	This parameter specifies the directory where the software writes timing reports that are generated after timing optimization. The default directory is `./timingReports` *Data_type*: string, optional
`-report_prefix` *file_name_prefix*	Specifies a prefix for `optDesign` report file names. *Default*: `DesignName_DesignStage` , where `DesignStage` is pre-CTS, post-CTS, or post-Route *Data_type*: string, optional
`-timing_debug_report`	When specified, generates a machine-readable timing debug report after completing CTS. This report is used for debugging timing results using the timing debug feature in Innovus. The number of paths to be reported can be specified using the `-num_paths` parameter. *Data_type*: bool, optional

Examples

- The following command runs a complete CCOpt optimization for the podV2 flow:
  `clock_opt_design`

Related Information

- `create_clock_tree_spec`
- `ccopt_design`
- `opt_design`
- `place_opt_design`
- The following topic in the *Innovus Stylus Common UI User Guide.*
  - Flow and Quick Start for CTS

clock_post_route_repair

```
clock_post_route_repair
[-help]
[-enable_average_id_reduction {true | false}]
[-enable_downsizing_pass {true | false}]
[-enable_drv_repair {true | false}]
[-enable_drv_repair_by_buffering {true | false}]
[-enable_refine_place {true | false}]
[-enable_routing_eco {true | false}]
[-enable_skew_repair {true | false}]
[-enable_skew_repair_by_buffering {true | false}]
```

Repairs clock tree design rule violations (DRVs) and skews that have occurred in the clock tree across a routing step. DRVs and skews can be fixed either by sizing existing clock instances or by adding buffers. Use this command to enable or disable average insertion delay reduction, global cell downsizing pass, refine placement of design, ECO routing, legalizer sharing, and the timing update step.

This command can be run at the following stages:

- After `ccopt_design` – when only the clock is routed
- After `route_design` , but before `opt_design -post_route` – when both clock and datapath are routed.

By default a post `ccopt_design` call to this command will fix DRVs and skew violations by resizing instances. It will also fix persistent DRVs by adding buffers. However, when called post `route_design`, this command is less disruptive by default and will only

perform resizing to fix DRVs. The `clock_post_route_repair` log will report a list of "active optimizations" for clarification of what steps will be executed.

By default, the command will modify the clock to reduce violations and then call `place_detail` and `route_eco` to patch up the modified routes.
Both `place_detail` and `route_eco` parts may be disabled to reduce runtime for user experimentation. But they will need to be called eventually, either with a final call to `clock_post_route_repair` or manually. Without them, timing measurements will be inaccurate.

Note:

- This command is integrated within the `ccopt_design` command and is run after clock routing. It can be identified in the `ccopt_design` log with a search for "PostConditioning....".
- This command is integrated within the `opt_design -post_route` command. It can be identified in the `opt_design` log with a search for "PRO...".
- If this command is called after running `ccopt_design` (when only the clock is routed), it will also call `route_early_global` on the datapath nets.
- The `clock_post_route_repair` command and the clock DRV fixing performed by running `opt_design -post_route` do not work when signoff effort-level for extraction is specified using `extract_rc_effort_level signoff` attribute. The signoff effort-level for extraction should be used only for ECO flows, after post_route optimization.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `clock_post_route_repair` parameter. For a detailed description of the command and all of its parameters, use the man command: `man clock_post_route_repair`.
`-enable_average_id_reduction{true | false}`	Enables or disables average insertion delay reduction. *Data_type*: enum, optional
`-enable_downsizing_pass {true | false}`	Enables or disables a global cell downsizing pass. *Data_type*: enum, optional
`-enable_drv_repair {true | false}`	Enables or disables DRV fixing. *Data_type*: enum, optional
`-enable_drv_repair_by_buffering {true | false}`	Enables or disables DRV fixing by adding buffers. *Data_type*: enum, optional
`-enable_refine_place {true | false}`	Enables or disables refine placement of design. Disabling this may result in instance overlaps. *Data_type*: enum, optional
`-enable_routing_eco {true | false}`	Enables or disables the ECO routing step, which corrects routing for new instance locations. Disabling this will leave inaccurate routing in place and lead to timing inaccuracy. *Data_type*: enum, optional
`-enable_skew_repair {true | false}`	Enables or disables fixing of clock tree skew. **Note the following**: • When `-enable_skew_repair` is set to `false` then whether the `-enable_skew_repair_by_buffering` parameter is specified as `true` or `false`, there is no skew fixing during `clock_post_route_repair`. • When `-enable_skew_repair` is set to `true`, while `-enable_skew_repair_by_buffering` parameter is set to `false`, the software resizes cells to fix skew. • When `-enable_skew_repair` is set to `true`, while `-enable_skew_repair_by_buffering` parameter is set to `true`, the software not only resizes cells but also does buffering to fix skew. *Data_type*: enum, optional
`-enable_skew_repair_by_buffering {true | false}`	Enables or disables fixing of clock tree skew by adding buffers. This will only help with lengthening short paths. *Data_type*: enum, optional

Related Information

- `ccopt_design`
- `opt_design`
- `place_detail`
- `route_eco`
- `route_early_global`
- Clock Tree Synthesis chapter of *Innovus Stylus Common UI User Guide*

commit_clock_tree_route_attributes

```
commit_clock_tree_route_attributes
[-help]
[-verbose]
```

The command traces the clock tree as defined by the CTS spec and sets net attributes to allow early-global-route to more predictably model clock nets. It sets isClock and isCTSClock on clock tree nets in the database. It applies routing preferences such as: `top_preferred_layer`, `bottom_preferred_layer`, `non_default_rule`, and `shield_net` based on the “route_type” CTS properties.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `commit_clock_tree_route_attributes` parameter. For a detailed description of the command and all of its parameters, use the man command: `man commit_clock_tree_route_attributes`.
`-verbose`	Prints the layer constraint summary for CCOpt nets after attribute application. *Data_type*: bool, optional

Related Commands

- `create_clock_tree_spec`
- `create_clock_tree`

connect_clock_tree_mesh_drivers

```
connect_clock_tree_mesh_drivers
[-help]
[-pin pin_names]
[-source_group source_group_name]
```

Uses the specified set of drivers and merges the nets at their outputs to form a single multi-driven net. To run this command successfully, one of the two options, `-pin` or `-source_group` must be specified.

It is possible that some or all of the driver pin nets are already multi-drive. You might be repeating the same command, or you could be merging additional drivers into an already multi-drive net. Running the command twice does nothing the second time. If the net of any driver pin is *already* multi-drive, the pin list will be interpreted to include all drivers of that net.

The command returns the resulting merged net through a Tcl return value.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `connect_clock_tree_mesh_drivers` parameter. For a detailed description of the command and all of its parameters, use the man command: `man connect_clock_tree_mesh_drivers`.
`-pin pin_names`	Specifies a list of driver output pins whose nets will be merged to form a multi-drive mesh net. *Data_type*: (pin)+, optional
`-source_group source_group_name`	Specifies the source group for the list of driver output pins. Only the root pins of the specified source group will be included in the list of driver output pins. *Data_type*: string, optional

Related Command

- `add_clock_tree_source_group_roots`

convert_lib_clock_tree_latencies

```
convert_lib_clock_tree_latencies
[-help]
[-latency_file_prefix prefix_name]
[-pins pin_list]
[-views view_list]
[-override_existing_latencies | -sum_existing_latencies]
[-sum_existing_latencies | -override_existing_latencies_pins pin_list]
[-override_existing_latencies | -sum_existing_latencies_pins pin_list]
[-sum_existing_latencies | -sum_existing_latencies_pins pin_list]
[-override_existing_latencies | -override_existing_latencies_pins pin_list]
```

Converts the Liberty `max_clock_tree_path` and `min_clock_tree_path` (MCTP) data to per-pin clock latency adjustments.

Note:

- Run the command before running the `create_clock_tree_spec` command so that the the output of `convert_lib_clock_tree_latencies` command impacts the clock tree specification that is created.
- Before running this command, ensure that the analysis type is set as `on-chip variation` (OCV) in the following manner:

  ```
  set_db timing_analysis_type ocv
  ```

By default, the conversion will consider all instance pins with MCTP data in all analysis views. Pins which have any existing network latency, differing from the corresponding clock network latency will be skipped completely for all clocks and for all views – this avoids the conversion process from interfering with any SDC-specified pin latencies. Alternatively, existing latency offsets can be summed or overridden for all or names pins using the appropriately named override and sum option. Additionally, the `-pins` option can be used to consider only a specified list of pins instead of all pins with MCTP data.

By default, the conversion will internally apply the resulting `set_clock_latency` commands to the in memory timing constraints. Alternatively the `-latency_file_prefix` parameter can be specified to have one file per analysis view written containing the `set_clock_latency` commands.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `convert_lib_clock_tree_latencies` parameter. For a detailed description of the command and all of its parameters, use the man command: `man convert_lib_clock_tree_latencies`.
`-latency_file_prefix` *`prefix_name`*	
	When specified, the software, instead of applying changes, writes the commands to one latency file per analysis view. Any pre-existing files are overwritten. For example, `<`*`prefix`*`><`*`analysis_view_name`*`>.sdc` **Note**: The prefix may contain directory name components. New directories should be created, as required. *Data_type*: string, optional
`-override_existing_latencies`	
	When specified, includes all pins with existing latency offsets and overwrites the existing latency offsets. **Note**: • The `-override_existing_latencies` and `-sum_existing_latencies` parameters are mutually exclusive. • The `-override_existing_latencies` and `-sum_existing_latencies_pins` parameters are mutually exclusive. • The `-override_existing_latencies` and `-override_existing_latencies_pins` parameters are mutually exclusive. *Data_type*: bool, optional
`-override_existing_latencies_pins` *`pin_list`*	

	Includes only the specified list of pins with existing latency offsets and overwrites the existing latency offsets. **Note**: • The `-override_existing_latencies_pins` and `-override_existing_latencies` parameters are mutually exclusive. • The `-override_existing_latencies_pins` and `-sum_existing_latencies` parameters are mutually exclusive. • The software issues an error if a pin is specified in both `-override_existing_latencies_pins` and `-sum_existing_latencies_pins` parameters. *Data_type*: (pin)+, optional
`-pins` *`pin_list`*	
	Specifies the clock endpoint pins to consider. By default, the software considers all clock endpoints with the library `max/min_clock_tree_path` attributes. **Note**: The software issues an error if a pin is specified in either the `-sum_existing_latencies_pins` or the `-override_existing_latencies_pins` list, but the pin is not included in the list of pins specified using the `-pins` parameter. *Data_type*: (pin)+, optional
`-sum_existing_latencies`	
	When specified, includes all pins with existing latency offsets and sums the existing latency offset with the library-specified offset. **Note**: • The `-sum_existing_latencies` and `-override_existing_latencies` parameters are mutually exclusive. • The `-sum_existing_latencies` and `-override_existing_latencies_pins` parameters are mutually exclusive. • The `-sum_existing_latencies` and `-sum_existing_latencies_pins` parameters are mutually exclusive. *Data_type*: bool, optional
`-sum_existing_latencies_pins` *`pin_list`*	
	Includes only the specified list of pins with existing latency offsets and sums the existing latency offset with the library-specified offset. **Note**: • The `-sum_existing_latencies_pins` and `-sum_existing_latencies` parameters are mutually exclusive. • The `-sum_existing_latencies_pins` and `-override_existing_latencies` parameters are mutually exclusive. • The software issues an error if a pin is specified to both `-sum_existing_latencies_pins` and `-override_existing_latencies_pins` parameters. *Data_type*: (pin)+, optional
`-views` *`view_list`*	
	Specifies the analysis views to operate on. By default the software operates on all active analysis views. *Data_type*: string, optional

Examples:

Example 1 – Two clocks, existing clock network latency, no pre-existing latency override

```
Circuit

ck1 ---> +------+

         | mux | -----> icg --+--> flop1/CK   (regular flop pin without MCTP data)

ck2 ---> +------+              +--> macro1/CK  (pin with MCTP data, of 0.222R/0.232F at 0.125 transition time)

                              +--> macro2/CK  (pin with MCTP data, of 0.333R/0.343F at 0.125 transition time)

SDC

  create_clock -period 1.0 [get_ports ck1]
```

```
  create_clock -period 1.0 [get_ports ck2]
  set_clock_transition 0.125 [get_clocks ck1]
  set_clock_transition 0.125 [get_clocks ck2]
  set_clock_latency 1.5 [get_clocks {ck1}]
  set_clock_latency 2.0 [get_clocks {ck2}]
Command and log output
convert_lib_clock_tree_latencies -latency_file_prefix lat_ -sum_existing_latencies -views default_analysis_view_setup
Converting library clock tree path delays to clock latencies for analysis_view default_analysis_view_setup
Found 2 of 3 clock endpoints with library clock tree path data
+-------------------------------------------------------------------+
| Clock endpoint         | Status                                   |
|------------------------+------------------------------------------|
| macro1/ck              | converted                                |
| macro2/ck              | converted                                |
+-------------------------------------------------------------------+
Writing latencies to file 'lat_default_analysis_view_setup.sdc'
Output file (the comments should not be written in the file)
  set_clock_latency 1.278 [get_pins {macro1/CK}] -clock [get_clocks {ck1}] -rise /1.5-0.222=1.278
  set_clock_latency 1.268 [get_pins {macro1/CK}] -clock [get_clocks {ck1}] -fall /1.5-0.232=1.268
  set_clock_latency 1.778 [get_pins {macro1/CK}] -clock [get_clocks {ck2}] -rise /2.0-0.222=1.778
  set_clock_latency 1.768 [get_pins {macro1/CK}] -clock [get_clocks {ck2}] -fall /2.0-0.232=1.768
  set_clock_latency 1.167 [get_pins {macro2/CK}] -clock [get_clocks {ck1}] -rise /1.5-0.333=1.167
  set_clock_latency 1.157 [get_pins {macro2/CK}] -clock [get_clocks {ck1}] -fall /1.5-0.343=1.157
  set_clock_latency 1.667 [get_pins {macro2/CK}] -clock [get_clocks {ck2}] -rise /2.0-0.333=1.667
  set_clock_latency 1.657 [get_pins {macro2/CK}] -clock [get_clocks {ck2}] -fall /2.0-0.343=1.657
```

Example 2 – User-specified pin list

```
Circuit
  ck1 --> icg --+--> flop1/CK   (regular flop pin without MCTP data)
                +--> macro1/CK  (pin with MCTP data, of 0.222R/0.232F at 0.125 transition time)
                +--> macro2/CK  (pin with MCTP data, of 0.333R/0.343F at 0.125 transition time)
SDC
  create_clock -period 1.0 [get_ports ck1]
  set_clock_transition 0.125 [get_clocks ck1]
Command and log output
convert_lib_clock_tree_latencies -latency_file_prefix lat_ -pins {flop1/CK macro2/CK flop3/D} -views default_analysis_view_setup
Converting library clock tree path delays to clock latencies for analysis_view default_analysis_view_setup
Found 2 of 3 clock endpoints with library clock tree path data
+-------------------------------------------------------------------+
| Clock endpoint         | Status                                   |
|------------------------+------------------------------------------|
| flop1/ck               | skipped, no library clock tree path data  |
| macro2/ck              | converted                                |
| flop3/D                | skipped, not a clock endpoint            |
+-------------------------------------------------------------------+
Writing latencies to file 'lat_default_analysis_view_setup.sdc'
```

```
Output file
  set_clock_latency -0.333 [get_pins {macro2/CK}] -clock [get_clocks {ck1}] -rise
  set_clock_latency -0.343 [get_pins {macro2/CK}] -clock [get_clocks {ck1}] -fall
```

Example 3 – Multiple analysis views

```
Command and log output
convert_lib_clock_tree_latencies -latency_file_prefix lat_ -pins {flop1/CK macro2/CK}
Converting library clock tree path delays to clock latencies for analysis_view 'setup1'
Found 2 of 3 clock endpoints with library clock tree path data
+-------------------------------------------------------------------+
| Clock endpoint         | Status                                   |
|------------------------+------------------------------------------|
| macro1/ck              | skipped, existing latency found          |
| macro2/ck              | converted                                |
  +-------------------------------------------------------------------+
Converting library clock tree path delays to clock latencies for analysis_view 'setup2'
Found 2 of 3 clock endpoints with library clock tree path data
+-----------------------------------------------------------------------+
| Clock endpoint         | Status                                       |
|------------------------+----------------------------------------------|
| macro1/ck              | skipped, existing latency found in other view |
| macro2/ck              | converted                                    |
+-----------------------------------------------------------------------+
Writing latencies to file 'lat_setup1.sdc'
Writing latencies to file 'lat_setup2.sdc'
```

Related Information

- get_lib_clock_tree_path_delay
- create_clock
- set_clock_latency
- The following topic in *Innovus Stylus Common UI User Guide*
  - Converting Library Path Delays to Clock Latencies

create_clock_tree

```
create_clock_tree
[-help]
[-name clockname]
[-no_skew_group]
-source pin
[-stop_at_sdc_clock_roots]
```

Defines a new clock tree within the design. You can define more than one clock tree and ordering of clock tree definitions is important. CCOpt can only optimize defined clock trees. When you run this command, CCOpt runs an algorithm known as clock tree extraction. Clock tree extraction examines the design starting at the root of a clock tree and decides which pins in the circuit are clock sinks for that tree and its clock gating and logical structure.

The `create_clock_tree` command tells CCOpt that a part of the circuit is a clock tree. CCOpt then synthesizes the clock tree when you run the `ccopt_design` command. CCOpt treats clock trees and datapaths differently for power analysis purposes.

By default, defining a clock tree also creates associated skew groups.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `create_clock_tree` parameter. For a detailed description of the command and all of its parameters, use the man command: `man create_clock_tree`.
`-name clockname`	
	Specifies the user-defined identifier for the clock tree. This name should be unique. If this parameter is not specified, the software uses the name of the source pin by default. *Data_type*: string, optional
`-no_skew_group`	
	Specifies that no default skew group will be defined for the sinks in the clock tree. If any skew groups are required, they will need to be defined manually using the `create_skew_group` command. *Data_type*: bool, optional
`-source pin`	
	Specifies the name of the source pin of the clock tree in the design. *Data_type*: (pin|port), required
`-stop_at_sdc_clock_roots`	
	Specifies that clock tree definition should stop searching for parts of the clock tree through the root of SDC clocks. *Data_type*: bool, optional

Example

The following command creates a clock tree named "ck1" by tracking the clock path from the pin "clk":

```
create_clock_tree -name ck1 -source clk
```

Related Information

- `create_skew_group`
- `ccopt_design`
- The following section in Clock Tree Synthesis chapter of *Innovus Stylus Common UI User Guide*
  - Clock Tree Concepts and Specifications

create_clock_tree_source_group

```
create_clock_tree_source_group
[-help]
-clock_trees clock_tree_list
-name clock_tree_source_group_name
```

Creates a clock tree source group. These source groups are used when a clock tree has multiple equivalent roots, and you want CTS to automatically swap sinks between roots to reduce insertion delay. It is required for flows where we are connecting one logical clock tree to multiple tap-off points from a top level H-tree or a mesh.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `create_clock_tree_source_group` parameter. For a detailed description of the command and all of its parameters, use the man command: man `create_clock_tree_source_group`.
`-clock_trees` *clock_tree_list*	Specifies a Tcl list of clock tree names that should be grouped together. CTS will attempt to swap sinks between these clock trees in order to balance the insertion delay under each clock tree's root and may clone clock logic to help this. This parameter is required. **Note:** This Tcl list is the same as that defined previously using the `create_clock_tree` command. *Data_type*: string, required
`-name` *clock_tree_source_group_name*	
	Specifies the user-defined name for the new clock tree source group. The name must be unique. This parameter is required. *Data_type*: string, required

Example

The following command creates a clock tree source group containing three clock trees: clk1, clk2 and clk3:

```
create_clock_tree_source_group -name clk_grp -clock_trees {clk1 clk2 clk3}
```

Related Information

- `add_clock_tree_source_group_roots`
- `assign_clock_tree_source_groups`
- `ccopt_design`
- `create_clock_tree_spec`
- `create_clock_tree`
- `delete_clock_tree_source_group`
- Clock Tree Synthesis chapter in the *Innovus Stylus Common UI User Guide*

create_clock_tree_spec

```
create_clock_tree_spec
[-help]
[-keep_all_sdc_clocks]
[-out_file filename]
[-views view_list]
```

Creates a clock tree network with associated skew groups and other clock tree synthesis (CTS) configuration settings such as ignore pins, case analysis, maxTrans, and so on based on a multi-mode timing configuration in the common timing engine (CTE).

When you run this command, one skew group will be created for each SDC clock in each constraint mode. Each such skew group will be able to take a skew target from any delay corner, but will have this target set to “ignore” unless there is an active analysis view that links the constraint mode for the skew group to that delay corner. The output of this command is a sequence of Innovus Tcl commands that can be edited manually.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `create_clock_tree_spec` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man create_clock_tree_spec`.
`-keep_all_sdc_clocks`	
	Specifies that all clocks and clock trees should be kept in 1:1 ratio, wherever possible. This means that CCOpt will maintain generated clock names for clock trees, even where these are functionally redundant, for example, clocks defined on buffer outputs. By default, CCOpt will not define these redundant clock trees. *Data_type*: bool, optional
`-out_file filename`	

	Writes this clock tree specification script file in Stylus Common UI format. This is an optional parameter. Providing the filename writes out the specified file. The file is not executed. To execute the file, run the following:
	`create_clock_tree_spec -out_file spec.tcl`
	`source spec.tcl`
	Note: You can execute the file only once. If you run the above again, the software gives an error message:
	`source spec.tcl`
	`Cannot run clock tree spec: clock trees are already defined.`
	The already loaded specification file can be removed by using either of the following commands:
	• `delete_clock_tree_spec`
	• `reset_cts_config`
	Note: If this parameter is not specified, the clock trees are defined by directly running clock tree extraction without the creation of a clock tree extraction script.
	Data_type: string, optional

`-views view_list`	
	Specifies the Tcl list of analysis view names. By default, the software considers all active analysis views. *Data_type*: (analysis_view)+, optional

Example

- The following example shows how running `create_clock_tree_spec` defines clock trees by directly running clock tree extraction instead of creating a clock tree extraction script:

```
get_db clock_trees
create_clock_tree_spec
get_db clock_trees
m_clk m_digit_clk m_dsram_clk m_ram_clk m_rcc_clk m_spi_clk refclk<1> refclk
```

Related Information

- `ccopt_design`
- `delete_clock_tree_spec`
- `get_db`
- `reset_cts_config`
- The following sections in Clock Tree Synthesis chapter of *Innovus Stylus Common UI User Guide*
  - The Clock Tree Synthesis Engines
  - Flow and Quick Start
  - Concepts and Clock Tree Specification

create_flexible_htree

```
create_flexible_htree
[-help]
[-adjust_sink_grid_for_aspect_ratio {true | false}]
-final_cell base_cell
[-image_directory dir_name]
[-hv_balance {true | false}]
[-inverting]
[-layer_density value]
[-max_driver_distance value]
[-max_root_distance value]
[-mode {drv | distance}]
-name flexibleHtreeName
[-omit_symmetry string]
[-partition_boundary_polarity {non_inverting | inverting | ignore}]
[-partition_groups {{partition ...} [max_boundary_net_length] [inverting | non_inverting] [subtree] ...}]
[-power_weight value]
[-sink_grid {columns rows}]
[-sink_grid_box {xmin ymin xmax ymax}]
[-sink_grid_exclusion_regions {{xmin ymin xmax ymax} ...}]
[-sink_instance_prefix prefixName]
[-sink_grid_sink_area {width height}]
[-sink_pins {{pin_name | {xmin ymin xmax ymax}} ...}]
-source  pin | port
[-stop_at_sdc_clock_roots]
-trunk_cell base_cell
```

Creates a design object for a flexible H-tree, under the specified pin. The H-tree is adjusted to consider placement and routing blockages, and any power routing.

You can specify this command in a sequence to create multiple flexible H-trees.

Note: This command does not synthesize flexible H-trees. After creating or defining flexible H-trees using this command, use the `synthesize_flexible_htrees` command to synthesize all the flexible H-trees created using this command.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `create_flexible_htree` parameter. For a detailed description of the command and all of its parameters, use the man command: `man create_flexible_htree`.
`-adjust_sink_grid_for_aspect_ratio {true | false}`	
	When set to true, adjusts the sink grid for aspect ratio. *Default*: `true` *Data_type*: string, optional
`-final_cell base_cell`	
	Specifies the library cell to use for the H-tree sinks. It must be a buffer or an inverter. *Data_type*: (base_cell), optional
`-image_directory dir_name`	

	Specifies the name of the directory to which the images generated by the H-tree synthesis algorithm are written. The images are PNG files. Color coding of images is provided below: White: Unobstructed edges of the synthesis grid Red: Grid points that are blocked for trunk cell placement in all modules Orange: Grid points that are blocked for final cell placement in all modules Red orange: Grid points that are blocked for trunk and final cell placement in all modules Yellow circle: The grid point of the source Yellow points: Candidate grid points of H-tree sinks Yellow rectangle: The sink area containing target grid point candidates of H-sinks, adjusted to the synthesis grid Brown: If specified, the sink grid box snapped to the synthesis grid Green/blue: The edges of the synthesized H-tree Purple: H-tree repeaters Cyan: Partition boundaries in hierarchical designs *Data_type*: string, optional
`-hv_balance {true | false}`	
	When set to `true`, specifies that the horizontal and vertical metal wires can only be balanced against other wires of the same orientation. When set to `false`, specifies that any wire can be balanced against any other wire. Default: `true` *Data_type*: string, optional
`-inverting`	Specifies whether the tree will invert its input or not. By default, this parameter is not enabled. *Data_type*: bool, optional
`-layer_density` *value*	
	Specifies the layer density used to compute the parasitics for timing estimates of H-tree nets. *Data_type*: float, optional, min=0, max=1
`-max_driver_distance` *value*	
	When specified, the software ignores DRVs and uses the given value as the maximum length of the nets connecting the H-tree drivers. *Data_type*: float, optional
`-max_root_distance` *value*	
	When specified, the software ignores DRVs and uses the given value as the maximum length of the net connecting the root and the first driver of the H-tree. This value can only be specified if the `-max_driver_distance` parameter is also specified. *Data_type*: float, optional
`-mode {drv | distance}`	
	Specifies the driver insertion mode. When set to `drv`, drivers are inserted to avoid DRVs. When set to `distance`, the parameters, `-max_driver_distance` and `-max_root_distance` determine the maximum net lengths allowed. Transitions and delays are not computed in this case. *Default*: `drv` **Note**: When this parameter is set to `distance`, the `-max_driver_distance` and `-max_root_distance` parameters must be specified. *Data_type*: string, optional
`-name` *flexibleHtreeName*	

	Specifies the name of the flexible H-tree to be created. This parameter is required. *Data_type*: string, optional
`-omit_symmetry` *string*	
	Specifies that no symmetry drivers and/or branches should be added to balance the pin load of nets in the flexible H-tree. Symmetry drivers are drivers that are added to balance pin capacitances at branch points. For example, if buffer pairs are inserted at branch points, one of these buffers may not drive any fanout and is inserted to match other buffer pairs at the same level of the flexible H-tree. Omitting symmetry drivers can reduce the power of the H-tree but increase its skew. Drivers that are added to terminate symmetry branches do not count as symmetry drivers and are never omitted. Similarly, symmetry branches are branches that are needed to balance the wire load at branch points. They are added to match other branchpoints at the same level of the flexible H-tree. Symmetry branches are always terminated with drivers, which are not counted as symmetry drivers. Possible values for this parameter are: `false`: Add symmetry branches and drivers `true`: Omit both symmetry drivers and branches `drivers`: Omit symmetry drivers `branches`: Omit symmetry branches `{drivers branches}`: Omit both symmetry drivers and branches `{}`: Add symmetry branches and drivers By default, symmetry branches and buffers are added. *Data_type*: string, optional
`-partition_boundary_polarity {non_inverting | inverting | ignore}`	
	Specifies the polarity of the clock signal with regard to the source of the flexible H-tree when entering partitions. The following values are allowed: • `non_inverting`: When specified, allows only non-inverting polarity at partition entry points. • `inverting`: When specified, allows only inverting polarity at partition entry points. • `ignore`: When specified, allows inverting and non-inverting polarity at partition entry points. *Default*: `non_inverting`
`-partition_groups {{`*partition* `...} [max_boundary_net_length] [inverting | non_inverting] [`*subtree*`] ...}`	

	Specifies the groups in which partitions are clustered in channelless designs. Nested lists imply allowed crossings between groups. Each group has zero or one input port and each partition must only be specified once. Optionally, a maximum pre-route net length from the boundary of a partition group can be specified. The maximum pre-route net length must be specified as the second parameter of a nested partition group. Optionally, the next argument specifies the clock phase when entering the partition group in relation to the root pin of the H-tree. Allowed values are 'inverting' and 'non_inverting'. If no value is specified, the clock phase is unconstrained. The clock phase is unconstrained when crossing the boundaries of partitions within the same group. All specified clock phases must be either inverting or non_inverting. For example, if the following partitions are specified: `{{A} {{C D} 50 non_inverting {{E F}} {{G}}}}` It means the following: • The H-tree starts in partition A and descends into group C/D. • From partition group C/D, the tree descends into groups E/F and G. • Any clustering of sinks inside the C/D and E/F groups is allowed, potentially crossing internal partition boundaries several times. • Partition A has no clock input port and one clock output port • Partition group C/D has one clock input port and two clock output ports • Partition group E/F and G have one clock input port and no clock output port • The maximum net length from the entry point into partition group C/D is 50um (pre-route) • The clock phase is non_inverting when entering parition group C/D. The clock phase is unconstrained when entering other partition groups. Default: `{}` *Data_type*: string, optional
`-power_weight` *value*	
	Specifies the power versus insertion delay trade-off. Valid values are between 0 and 1, specifying the weight that is put on power optimization during the synthesis of flexible H-trees. *Default*: `1` *Data_type*: float, optional, min=0, max=1
`-sink_grid {columns rows}`	
	Specifies the columns and rows of a grid of H-tree sinks. **Note**: This parameter is not supported in master/clone designs. *Data_type*: string, optional
`-sink_grid_box {xmin ymin xmax ymax}`	
	Specifies the area of the box that the grid of H-trees should cover. *Data_type*: string, optional
`-sink_grid_exclusion_regions {{`*xmin ymin xmax ymax*`} ...}`	
	Specifies regions from which sinks of the sink grid are excluded if the regions completely cover the respective sink area. *Data_type*: list, optional
`-sink_grid_sink_area {width height}`	
	Specifies the approximate size of the rectangle describing valid locations for final cells, specified using the `-final_cell` parameter, per H-tree sink in the grid. *Data_type*: string, optional
`-sink_instance_prefix` *prefixName*	
	Specifies the prefix used for instance names of final cells, specified using the `-final_cell` parameter and the sinks specified using the `-sinks` parameter. *Data_type*: string, optional
`-sink_pins {`*pin_name* `|` `{` *xmin ymin xmax ymax* `}}`	

	Specifies approximate rectangular sink regions for the buffer locations at the end of the tree, which are specified using the `-final_cell` parameter or pins. The specified pins must be in the clock tree. Nested lists imply clustering relationships. *Data_type*: list, optional
`-source pin | port`	
	Specifies the pin under which to create the flexible H-tree. This parameter is required. *Data_type*: (pin|port|hpin)+, required
`-stop_at_sdc_clock_roots`	
	Specifies that the clock tree definition should stop searching for parts of the clock tree at the root of SDC clocks.
`-trunk_cell` *base_cell*	Specifies the library cell to use inside the flexible H-tree. It must be either a buffer or an inverter. This parameter is required. *Data_type*: (base_cell), required

Examples

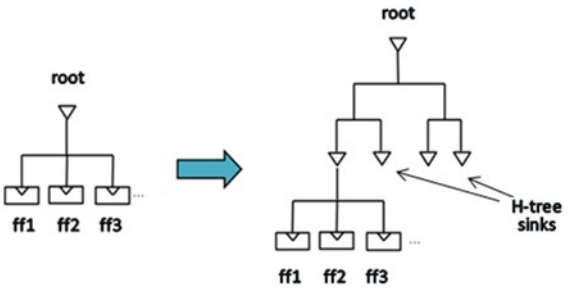
- The following command creates a flexible H-tree for four sinks under the the pin "`root/Y`". The sinks of the H-tree will be of type `BUFX16` and placed approximately inside the specified sink rectangles. Other buffers added to the H-tree will be of type `BUFX8`. No symmetry buffers are added to balance the pin load of nets. The flexible H-tree is then synthesized using the `synthesize_flexible_htrees` command.

```
create_flexible_htree -name tree0 -source root/Y -sink_pins { {200 200 300 300} {200 700 300 800} {700 200 800 300} {700 700 800 800}
} -final_cell BUFX16 -trunk_cell BUFX8 -omit_symmetry drivers
synthesize_flexible_htrees
```

- The following commands create two flexible H-trees under the pins "`rootA/Y`" and "`rootB/Y`". The sinks will be of the type `INVX8` and will be placed in the specified sink grids. Other buffers added to the H-trees will be of type `INVX12`. No symmetry buffers are added to balance the pin load of nets. The flexible H-trees are then synthesized using the `synthesize_flexible_htrees` command.

```
create_flexible_htree -name htreeA -source rootA/Y -sink_grid {2 2} -final_cell INVX8 -trunk_cell INVX12 -omit_symmetry drivers
create_flexible_htree -name htreeB -source rootB/Y -sink_grid {4 4} -final_cell INVX8 -trunk_cell INVX12 -omit_symmetry drivers
synthesize_flexible_htrees
```

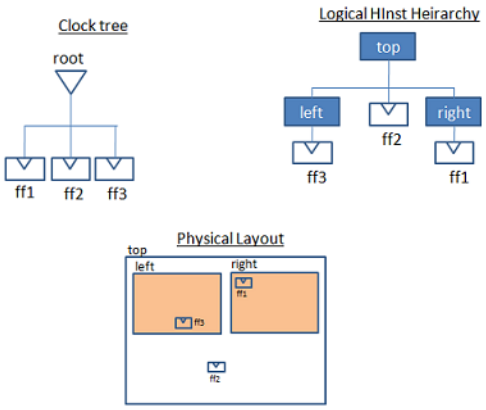
Note: During flexible H-tree synthesis a number of buffers (or inverters) are created. In designs with no fence regions, every instance in the flexible H-tree, including all H-tree sinks, are assigned to the top-level hInst. In this case, the `create_flexible_htree` command moves the original fanout of the root pin of the flexible H-tree to a single H-tree sink as shown in the image below.



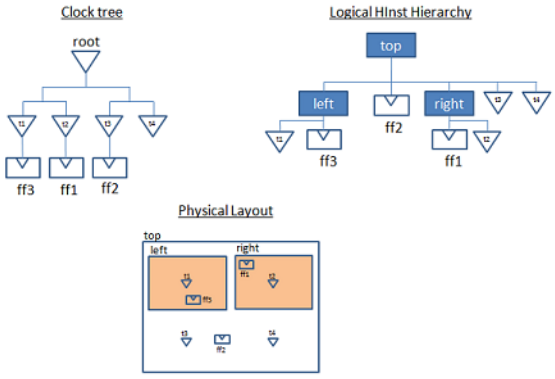
In designs where there are fence regions, the command chooses hInsts for instances in the flexible H-tree considering fence region constraints. The fanout of the root pin of the flexible H-tree is connected to H-tree sinks that best match the hInst to which a fanout instance belongs.

The images below illustrate a clock tree with three clock tree sinks, ff1, ff2, and ff3 and three hInsts, "top", "left", and "right". The hInsts "left" and "right" are fenced. Fenced regions are shown in pale orange. Sink ff1 is in hInst "right", ff2 is in the top-level hInst, and ff3 is in hInst "left". The hInsts "left" and "right" are children of the top-level hInst. After flexible H-tree synthesis there are four H-tree sink instances, t1, t2, t3, and t4 laid out in a square such that t1 is in the fence region for the hInst "left", t2 is in the fence region for the hInst "right", and the other two are in hInst "top". The flops have been connected to H-tree sinks that match their fence constraints. Note that when running `ccopt_design` after creating the flexible H-tree, multi-tap CTS may reconnect ff2 to t4 rather than t3.

Before H-tree Synthesis



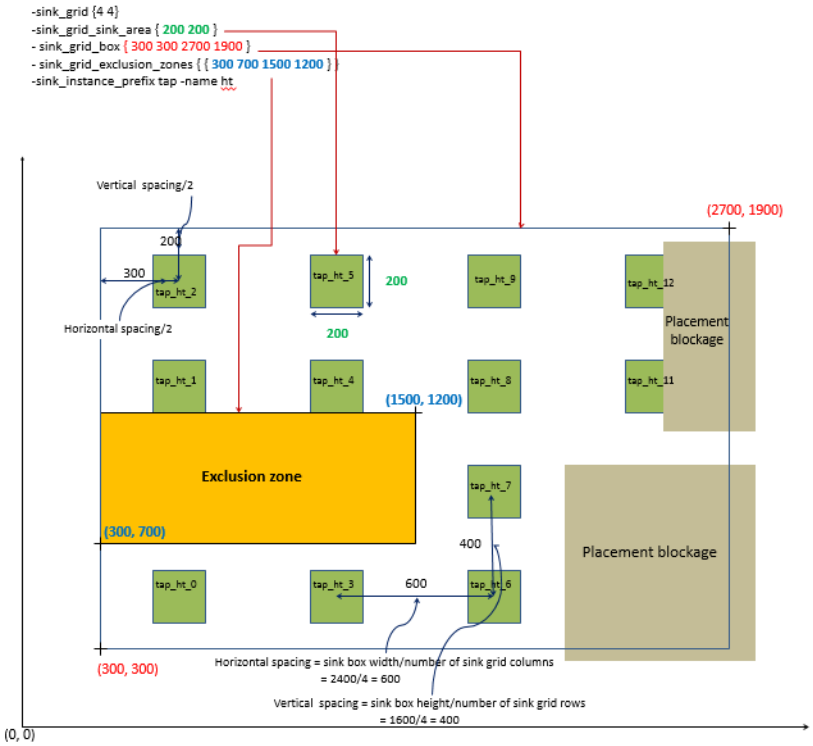
After H-tree Synthesis



Using the Sink Grid Parameters

The diagram below illustrates the usage of the sink grid parameters of this command. Note the following:

- The sink grid is specified as {4 4}, which means it will have four columns and four rows
- No H-tree sinks are created for sink areas that are completely covered by placement blockages or exclusions zones
- H-tree sink instances with the name specified in the diagram will be placed inside the unblocked parts of the sink areas
- The sink grid is adjusted by snapping it to the synthesis grid, which means that the specified sink areas are treated only as a guide



Related Information

- ccopt_design
- delete_flexible_htrees
- get_db
- set_db
- synthesize_flexible_htrees
- The following attributes:
  - cts_flexible_htree_placement_legalization_effort
  - cts_flexible_htree_partition_boundary_polarity
  - flexible_htrees

  For more information about CTS attributes, see the cts Category Attributes chapter.
- The following section in Clock Tree Synthesis chapter of *Innovus Stylus Common UI User Guide*
  - Flexible H-Tree and Multi-Tap Clock Flow
- For details of the use model for this feature, see the Flexible H-tree and Multi-Tap Clock Flow in Innovus Application Note on the Cadence Online Support website.

create_generated_clock_tree

```
create_generated_clock_tree
[-help]
[-generated_by pins]
[-name clockname]
[-parents parents]
[-search_method method]
-source pin
[-stop_at_sdc_clock_roots]
```

Defines a generated clock tree within the design. Use this command if you want to balance the sinks in the generated tree against those in the parent trees. CCOpt accomplishes this action by pulling up any sinks in the parent trees that contribute directly to the generated clock tree. The insertion delay under such sinks in the parent trees will be set to the insertion delay of the generated clock tree plus the delay between the sink pin and the source of the generated clock tree.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `create_generated_clock_tree` parameter. For a detailed description of the command and all of its parameters, use the man command: `man create_generated_clock_tree`.
`-generated_by pins`	Specifies the sink pins in the parent clock trees for which the delay between the parent and generated clock trees should be considered. If this parameter is not specified, all possible timing paths between sinks in the parent tree and the source of the generated clock tree will be considered. This parameter is mutually exclusive with the `-search_method` parameter. *Data_type*: (pin|port)+, optional
`-name clockname`	Specifies the user-defined identifier for the generated clock tree. This name should be unique. If this parameter is not specified, the software uses the name of the source pin by default. *Data_type*: string, optional
`-parents parents`	Specifies the parent clock trees for the generated clock trees. This parameter should be specified if the parents are ambiguous. *Data_type*: string, optional

`-search_method {same_instance multi_stage}`	Specifies the search methdology to be used when tracing the path to the generator inputs.
	`multi_stage`
	This is the default method, and will select the current behavior of the software. When this method is specified, the software searches backwards through all the delay paths leading into the clock root pin, looking for pre-existing clock sinks. Every such clock sink found is a candidate generator input for the generated clock tree. This includes clock sinks that lie on different instances to the root pin - where multiple stages lie between the clock sink and the root pin. Additional filtering on the candidate generator inputs is applied by the `-parents` parameter to avoid producing cycles in the clock tree structure. Once the set of generator inputs is finalized, the delay path between each finalized generator input and the clock root pin is imported into the CTS clock tree data stuctures. If the generator input lies on a different instance of the clock root pin, ClockGeneratorPathNode objects are created.
	`same_instance`
	This method is similar to the `multi_stage` method, except that the software restricts its search to consider only clock sinks that lie on the same instance as the generated clock tree's root pin. This means that the software does not traverse net arcs looking for generator input clock sinks. Consequently, ClockGeneratorPathNode objects will not be needed to model any generator path.
	Note:
	• This functionally is identical to specifying the `-generated_by` parameter that lists all the clock sinks on the same instance as the clock root pin. However, it avoids the need to explicitly identify those clock sinks.
	• Irrespective of the search method, if no candidate generator clock sink is found, or if there are no valid generator inputs discovered in the set of clock sinks, creation of the generated clock tree fails with an error.
	The software uses `multi_stage` method by default.
	This parameter is mutually exclusive with the `-generated_by` parameter.
	Data_type: enum, optional

`-source pin`	Specifies the name of the source pin of the generated clock tree. This is a required parameter. *Data_type*: (pin|port), required
`-stop_at_sdc_clock_roots`	Specifies that clock tree definition should stop searching for parts of the clock tree through the root of the SDC clocks. *Data_type*: bool, optional

Example

- The following command defines a generated clock tree `clk_div_2`, which is generated from the `gen_clk/CK` clock pin, using the `gen_clk.QN` pin as the source pin:
```
create_generated_clock_tree -name clk_div_2 -source gen_clk/QN -generated_by gen_clk/CK
```

Related Information

- `ccopt_design`
- `create_clock_tree`
- `create_clock_tree_spec`
- The following sections in Clock Tree Synthesis chapter of *Innovus Stylus Common UI User Guide*
  - Automatic Clock Tree Specification Creation
  - Manual Setup and Adjustment of the Clock Specification

create_route_type

```
create_route_type
[-help]
[-em_route_rule rule_name]
[-em_route_rule_distance float]
[-min_stack_layer layer]
-name string
[-one_side_spacing_range bottomLayerNum:topLayerNum]
[-prefer_multi_cut_via]
[-shield_side one_side | both_side]
[-shield_tap_instance_insertion_effort {none | standard | high}]
[-stack_distance float]
[[-route_rule ndr_name] [-shield_net net_name] [-top_preferred_layer layer] [-bottom_preferred_layer layer] [-preferred_routing_layer_effort
{low | medium | high}]
[-mask mask_number [-layer_mask_range bottomLayerNum:topLayerNum]]
```

Creates a new route type and sets the routing properties for the nets. This command lets you create the **standard route_type** that defines a collection of routing attributes such as non-default rule (NDR), shield net, preferred layers and so on that can then be applied to either all leaf or all trunk nets for a particular clock tree. The type of route_type object created depends on the parameters you specify with this command. A standard route_type object is created if you specify the `-route_rule, -shield_net, top_preferred_layer` or `-preferred_routing_layer` parameters.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `create_route_type` parameter. For a detailed description of the command and all of its parameters, use the man command: man `create_route_type`.
`-bottom_preferred_layer layer`	Specifies the preferred lowest routing layer. This attribute is a soft limit, which means, NanoRoute might use a layer below the specified layer if necessary to complete routing. A layer name can be any real layer name, for example, from LEF or OpenAccess. Routing layer names can also be abbreviated as 1 for the first routing layer, 2 for the second routing layer and so on. Note, the software will check for real layer names first. So, if 1 matches a real layer name it might not be the first routing layer. If not specified, the router will freely route the net starting on the first routing layer, with no preference for higher layers. *Data_type*: (layer), optional
`-em_route_rule rule_name`	Specifies the EM non-default rule name. *Data_type*: string, optional
`-em_route_rule_distance float`	Specifies the distance from EM NDR rule source to sink. *Data_type*: float, optional
`-layer_mask_range bottomLayerNum:topLayerNum`	Specifies the layer range that the mask constrain should be applied on. For example, 3:4 means that the constrain should be applied on metal3 and metal4. *Data_type*: string, optional
`-mask mask_number`	Specifies the mask constraint number in allowed range (min=0, max=3). *Data_type*: int, optional
`-min_stack_layer layer`	Specifies that the output pins of the net use stacked vias from the pin to the given `layer` and the rest of the connections start from the mentioned `layer`. *Data_type*: (layer), optional
`-name string`	Specifies the name of the route type object. *Data_type*: string, required
`-one_side_spacing_range bottomLayerNum:topLayerNum`	Specifies the layer range that one side spacing constrain should be applied on. By default, NDR spacing is applied to both sides of an NDR net or wire. Use this option to specify the layer range for wires that have only one neighboring wire with minimum spacing, which means only one side needs to follow larger NDR spacing. *Data_type*: string, optional

`-prefer_multi_cut_via`	Specifies whether or not to use the multi-cut vias for connections at the net driver pins. *Data_type*: bool, optional
`-preferred_routing_layer_effort {low | medium | high}`	Specifies whether the preferred routing layer effort will be low, medium, or high. This parameter determines how flexible the NanoRoute router is when setting layer limits for routing specified nets. Use this parameter with the `-top_preferred_routing_layer` and the `-bottom_preferred_routing_layer` parameters. Specify `medium` or `high` routing layer effort level to make NanoRoute less flexible in setting layer limits for routing specified nets. NanoRoute will obey the preferred routing layer range more strictly. Specify `low` to make NanoRoute more flexible in setting layer limits for routing specified nets. *Default*: `high` *Data_type*: enum, optional
`-route_rule` *ndr_name*	Specifies the non-default routing (NDR) rule to associate with this route_type. *Default*: no NDR rule *Data_type*: string, optional
`-shield_net` *net_name*	Specifies a special net to use as a shield for a critical or high-speed net. Typically, shielded nets are routed before routing other nets. You can specify one or two shield nets, however, you cannot control which side is used by the special net that is routed on the same layer as the signal routing. The shield terminates near the pin. If not specified, the net is not shielded. *Data_type*: string, optional
`-shield_side` *one_side* | *both_side*	Specifies whether one or both sides of the shield is used. *Data_type*: enum, optional
`-shield_tap_instance_insertion_effort {none | standard | high}`	Specifies the clock route type for clock shielding of the high and low frequencies through the `high` and `standard` values respectively. The main difference between these values is related to the ground that the tie cell instance/via uses to achieve shield segments of the electrical connection. *Data_type*: enum, optional
`-stack_distance` *float*	Specifies that the cut distance of cuts on adjacent layers in the stacked vias are defined in `-min_stack_layer`. *Data_type*: float, optional
`-top_preferred_layer` *layer*	Specifies the top-preferred layer for the route type. *Data_type*: (layer), optional

Examples

- For example, use the following commands to create the standard route_types that specify the routing attributes we want to assign to particular nets:
```
create_route_type -name RT1 -route_rule NDR1
create_route_type -name RT2 -route_rule NDR2
create_route_type -name RT3 -route_rule NDR3 -top_preferred_layer M4 -bottom_preferred_layer M3
create_route_type -name RT4 -route_rule NDR4 -preferred_routing_layer_effort high -top_preferred_layer M4 -bottom_preferred_layer M3
```
- Use the following command to create the advanced route_type that will map clock tree nets to route_types:
```
create_route_type -name advRT -table { {0 500} {0 10} {RT1 RT2 RT3 RT4} }
```

create_skew_group

```
create_skew_group
[-help]
[-constrains cts | ccopt_initial | ccopt]
[-from_clocks clock_names]
[-from_constraint_modes constraint_mode_names]
[-from_delay_corners delay_corner_names]
-name skew_group_name
[-rank rank]
[-sinks pins | -shared_sinks pins | -exclusive_sinks pins | -auto_sinks | -filtered_auto_sinks pins | -balance_skew_groups skew_groups]
[-sources pins | -balance_skew_groups skew_groups]
[-target_insertion_delay value]
[-target_skew value]
```

Creates a new skew group within the design. CCOpt balances clock trees according to skew group requirements. Balancing between multiple trees can be achieved using the `-sources` parameter. The sinks to be balanced can either be specified explicitly using the `-shared_sinks` or `-exclusive_sinks` parameters or can be automatically inferred by using the `-auto_sinks` parameter.

Every skew group is assigned a rank, accessible through the `cts_exclusive_sinks_rank` attribute:

- **Shared skew groups** always have a `cts_exclusive_sinks_rank` value of zero.
- **Exclusive skew groups** always have a `cts_exclusive_sinks_rank` value greater than zero. When an exclusive skew group is created, CCOpt assigns that exclusive skew group a rank one greater than the highest existing skew group.

For example, consider the following sequence of commands:

1. `create_skew_group -name SG1 -sources top -shared_sinks [get_ccopt_clock_tree_sinks */CK]`
2. `create_skew_group -name SG2 -sources top -exclusive_sinks [get_ccopt_clock_tree_sinks *XYZ*/CK]`
3. `create_skew_group -name SG3 -sources top -exclusive_sinks [get_ccopt_clock_tree_sinks *XYZ_01*/CK]`

After running the first command, a single skew group `SG1` is created. This is a shared skew group, so has an `exclusive_sinks_rank` value of zero. All the "`CK`" pins in the design are members of this skew group. In addition, all the "`CK`" pins are active sinks in skew group `SG1`. This is because `SG1` is the highest ranked skew group so far, even though it has a rank of zero.

The second command defines an exclusive skew group `SG2`. This is given an `exclusive_sinks_rank` value of 1, which is one higher than the current highest rank of 0. Sinks that match the pattern `*XYZ*/CK` are now members of both `SG1` and `SG2`. However, they are only active sinks in `SG2`, which is the highest ranked parent skew group till now. Sinks that don't match this pattern remain active pins in `SG1`.

The third command defines another exclusive skew group, `SG3`. This is given an `exclusive_sinks_rank` value of 2, which is one higher than the current highest rank of 1. Sinks that match the pattern `*XYZ_01*/CK` (which must also be members of `SG2`, since those sinks would also match the pattern `*XYZ*/CK`) are now members of skew groups `SG1`, `SG2`, and `SG3`. However, they are only active sinks in `SG3`, which is the highest ranked parent skew group. Sinks that matched the pattern `*XYZ*/CK` but not `*XYZ_01*/CK` are members of both `SG1` and `SG2` but only active sinks of `SG2`. Sinks that don't match the pattern `*XYZ*/CK` are members of `SG1` and active sinks in `SG1`.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `create_skew_group` parameter. For a detailed description of the command and all of its parameters, use the man command: `man create_skew_group`.
`-auto_sinks`	When specified, determines the target sinks for the skew group based on the input pins reachable from the source. *Data_type*: bool, optional
`-balance_skew_groups skew_groups`	
	Specifies that the new skew groups should be created by merging the specified skew groups in order to balance them together as a single skew group. **Note**: The sources of skew groups are specified using the `-sources` parameter. However, if the `-sources` parameter is not specified and you are using the `-balance_skew_groups` parameter, then the sources are the sources of the skew groups that are specified using this parameter. *Data_type*: string, optional
`-constrains cts | ccopt_initial | ccopt`	

	Specifies whether the skew group should constrain `cts, ccopt_initial`, or `ccopt`. If the value includes cts, the created skew group will constrain `cts`, if the value includes `ccopt_initial`, the created skew group will be used to constrain CCOpt's initial solution.If the value includes `ccopt`, the created skew group will constrain CCOpt's final solution. *Data_type*: string, optional
`-exclusive_sinks` *pins*	
	Specifies the input pins in the design that CCOpt should consider as members of the new skew group. This will give the new skew_group an `exclusive_sinks_rank` 1 greater than any other currently defined skew group, effectively removing the pins from all skew_groups that they currently belong to. If both "`shared_sinks`" and "`exclusive_sinks`" are not specified, CCOpt will use the set of sinks reachable from the source if the `auto_sinks` switch is provided, or no sinks otherwise. *Data_type*: string, optional
`-filtered_auto_sinks` *pins*	
	Specifies that the target sinks should be determined automatically from the subset provided. The target sinks for the skew group are determined based on the input pins reachable from the source. However, of these pins, only those are selected that are also present in the list of pins provided as a filter. *Data_type*: string, optional
`-from_clocks` *clock_names*	
	Specifies the names of the clocks for which this skew group was extracted. *Data_type*: string, optional
`-from_constraint_modes` *constraint_mode_names*	
	Specifies the constraint mode for which this skew group was extracted. *Data_type*: string, optional
`-from_delay_corners` *delay_corner_names*	
	Specifies the delay corners associated with the constraint modes for which this skew group was extracted. *Data_type*: (delay_corner)+, optional
`-name` *skew_group_name*	
	Specifies the user-defined name for the new skew group. The name must be unique. *Data_type*: string, required
`-rank` *rank*	Specifies the exclusive_sinks_rank for this skew group. *Data_type*: int, optional
`-shared_sinks` *pins*	
	Specifies the input pins in the design that CCOpt should consider as members of the new skew group but are also shared with other skew groups. These pins are members of both the original skew group and the new skew group. This implies a `exclusive_sinks_rank` of 0. *Data_type*: string, optional
`-sinks` *pins*	Specifies the input pins in the design that CCOpt should consider as members of the new skew group. *Data_type*: string, optional
`-sources` *pins*	Specifies the source pins of the skew group. *Data_type*: string, optional
`-target_insertion_delay` *value*	
	Specifies the target insertion delay to be used for the skew group. CCOpt attempts to keep the latency of all sinks within half the target skew on either side of this delay. For example, `create_skew_group -name b1 -sources b1 -exclusive_sinks {A/CK B/CK G/CK} -target_insertion_delay 0.3ns -target_skew 0.1ns` *Data_type*: string, optional
`-target_skew` *value*	

	Specifies the skew that the clock tree synthesis should aim for. It sets the cts_`target_skew` attribute for the created skew group. If this parameter is not set, the software will use the default value that it gets from `cts_target_skew` attribute. For more information, see the cts Category Attributes chapter. *Data_type*: string, optional

Example

- The following command creates the specified skew group, `clk_disable`, for the specified source and exclusive sinks:

```
create_skew_group -name clk_disable -sources clk -exclusive_sinks {mod/ff1/CK mod/icg/CK}
```

Related Information

- `delete_skew_groups`
- `get_db`
- `get_skew_group_path`
- `get_skew_group_delay`
- `report_skew_groups`
- `set_db`
- `update_skew_group`
- The following sections in Clock Tree Synthesis chapter of *Innovus Stylus Common UI User Guide*
  - Clock Trees and Skew Groups
  - Automatic Clock Tree Specification Creation
  - Manual Setup and Adjustment of the Clock Specification

cts Category Attributes

- clock_trees
- clock_tree_source_groups
- cts_add_port_driver_max_distance_from_port
- cts_add_port_driver_max_distance_from_port_rows
- cts_adjacent_rows_legal
- cts_allow_non_std_clock_gate_inputs
- cts_balance_mode
- cts_blackbox_default_driver_base_pin
- cts_blackbox_default_load_base_pin
- cts_buffer_cells
- cts_buffer_cells_leaf
- cts_buffer_cells_top
- cts_cell_density
- cts_cell_halo_mode
- cts_cell_halo_rows
- cts_cell_halo_sites
- cts_cell_halo_x
- cts_cell_halo_y
- cts_clock_gating_cells
- cts_clock_gate_movement_limit
- cts_clock_source_cells
- cts_clock_tree_source_group_clock_trees
- cts_clock_tree_source_input_max_transition_time
- cts_cloning_inst_name_suffix
- cts_cloning_inst_name_suffix_source_group_assignment
- cts_delay_cells
- cts_detailed_cell_warnings
- cts_exit_if_no_placeable_area
- cts_exit_if_skew_target_over_constrained
- cts_exit_if_stage_delay_sigma_target_over_constrained
- cts_fix_clock_sinks
- cts_flexible_htree_enable_metrics_reporting
- cts_flexible_htree_partition_boundary_polarity
- cts_flexible_htree_placement_legalization_effort
- cts_force_target_max_transition_time
- cts_ignore_problematic_skew_as_result_of_dont_touch_nets
- cts_inst_name_prefix
- cts_inverter_cells
- cts_inverter_cells_leaf
- cts_inverter_cells_top
- cts_load_capacitance_cells
- cts_logic_cells

- cts_manage_power_intent_violations
- cts_max_fanout
- cts_max_instance_height
- cts_max_source_to_sink_net_length
- cts_max_source_to_sink_net_length_leaf
- cts_max_source_to_sink_net_length_top
- cts_max_source_to_sink_net_length_trunk
- cts_merge_clock_gates
- cts_merge_clock_logic
- cts_mixed_fanout_net_type
- cts_move_clock_gates
- cts_move_logic
- cts_net_name_prefix
- cts_override_vias
- cts_pin_insertion_delay_histogram_bin_size
- cts_post_route_enable
- cts_post_route_repair_drv
- cts_post_route_enable_routing_eco
- cts_post_route_repair_skew_by_buffering
- cts_primary_delay_corner
- cts_primary_reporting_skew_groups
- cts_primary_reporting_skew_groups_log_min_max_sinks
- cts_repair_congestion
- cts_repair_drv_by_buffering
- cts_report_skew_groups_only_with_targets
- cts_route_clock_tree_nets
- cts_route_type_leaf
- cts_route_type_top
- cts_route_type_trunk
- cts_routing_preferred_layer_effort
- cts_size_clock_gates
- cts_size_clock_sources
- cts_size_logic
- cts_skew_group_report_columns
- cts_skew_group_report_histogram_bin_size
- cts_spec_config_base_pin_trace_through_to
- cts_spec_config_create_clock_tree_source_groups
- cts_spec_config_create_generator_skew_groups
- cts_spec_config_create_generator_skew_groups_name_prefix
- cts_spec_config_create_reporting_only_skew_groups
- cts_spec_config_trace_multi_output_cells_with_single_clock_output
- cts_target_max_capacitance
- cts_target_max_capacitance_leaf

- cts_target_max_capacitance_top
- cts_target_max_capacitance_trunk
- cts_target_max_stage_delay_sigma
- cts_target_max_transition_time
- cts_target_max_transition_time_leaf
- cts_target_max_transition_time_top
- cts_target_max_transition_time_trunk
- cts_target_skew
- cts_timing_connectivity_based_skew_groups_balance_master_clocks
- cts_timing_connectivity_based_skew_groups
- cts_timing_connectivity_info
- cts_top_fanout_threshold
- cts_update_clock_latency
- cts_use_inverters
- cts_use_receiver_model_capacitance_for_drv
- flexible_htrees
- skew_groups

To set the value of any root attribute, use the `set_db` command:

```
set_db attribute_cts_library_trace_through_toname value
```

To get the current value of any root attribute, use the `get_db` command:

```
get_db attribute_name
```

To get the current value of all attributes in the `cts` category, use the following command:

```
get_db -category cts
```

Note: For attributes that are keyed, ensure that the design is loaded before using the `get_db attribute_name` command.

For more information about the CTS object attributes, see *Stylus Common UI Database Object Information*. The following categories of CTS objects are available:

- clock_tree
- clock_tree_source_group
- flexible_htree
- skew_group

clock_trees

```
clock_trees {list | clock_tree}
```

Default: `{}`

Data_type: string, read/write

Specifies a list of clock trees the pin is contained within. This includes parents of generated clock trees and all relevant parents when clock trees overlap.

Valid values: `list clock_tree`

clock_tree_source_groups

```
clock_tree_source_groups clock_tree_source_group_list
```

Default: `{}`

Data_type: string, read-only

Specifies the list of clock tree source groups to which this clock tree belongs.

Valid values: `clock_tree_source_group_list`

Read-only: This attribute cannot be modified by `set_db`.

cts_add_port_driver_max_distance_from_port

`cts_add_port_driver_max_distance_from_port` *`allowable_distance`*

Default: `auto`

Data_type: string, read/write

This attribute specifies the allowable distance, in `um` if no units suffix is specified, that a port driver is allowed to be placed from the port being buffered. If the driver cannot be added close enough to the port, it will not be added at all. This attribute, if set, overrides any value set using the attribute, `cts_add_port_driver_max_distance_from_port_rows`.

cts_add_port_driver_max_distance_from_port_rows

`cts_add_port_driver_max_distance_from_port_rows` *`allowable_cell_row_height_distance`*

Default: `50`

Data_type: integer, read/write

This attribute specifies the allowable distance, in standard cell row heights, that a port driver is allowed to be placed from the port being buffered. If the driver cannot be added close enough to the port it will not be added at all. The attribute `cts_add_port_driver_max_distance_from_port`, if set, overrides this value.

cts_adjacent_rows_legal

`cts_adjacent_rows_legal {true | false}`

Default: `false`

Data_type: string, read/write

Defines the clock halo in the y direction. If set to true, the y direction clock halo is zero and other clock instances are allowed in adjacent rows. The following attributes can be used to assign y direction clock halos within CTS:

- `cts_cell_halo_y`
- `cts_adjacent_rows_legal`
- `cts_cell_halo_rows`

Only one of these attributes is used to determine the clock halo in the y direction. The following rules determine which:

- If `cts_cell_halo_y` is set to a non-auto value, then this defines the y direction clock halo. The attributes `cts_adjacent_rows_legal` and `cts_cell_halo_rows` have no effect.
- If `cts_cell_halo_y` is set to auto and `cts_adjacent_rows_legal` is set to a non-auto value then `cts_adjacent_rows_legal` defines the clock halo in the y direction. The attribute `cts_cell_halo_rows` has no effect.
- If both `cts_cell_halo_y` and `cts_adjacent_rows_legal` are set to auto then `cts_cell_halo_rows` defines the clock halo in the y direction.

Valid values: `true false`

See also:
. `cts_cell_halo_y`
. `cts_cell_halo_rows`

cts_allow_non_std_clock_gate_inputs

`cts_allow_non_std_clock_gate_inputs {true | false}`

Default: `false`

Data_type: bool, read/write

If this attribute is set, CTS will allow the use of clock gates with non-standard pins. CTS considers the following pin types to be standard: clock pins, enable pins, test enable pins, retention pins, and power gating pins. Before starting clock tree synthesis, CTS will emit a warning, indicating which pin(s) it considers as non-standard.

cts_balance_mode

```
cts_balance_mode {full | cluster | trial}
```

Default: `full`

Data_type: string, read/write

Specifies the CTS mode setting to be used. If the default setting of `full` is not used, it causes CTS to halt before the final completion of the clock tree to facilitate clock tree inspection.

The possible values for this attribute are as follows:

- `full` - default value, a full CTS is performed.
- `cluster` - a cluster-only CTS is performed. The clock tree has no balancing delay applied.
- `trial` - The clock has only virtual (numeric annotation) balancing delays applied.

cts_blackbox_default_driver_base_pin

```
cts_blackbox_default_driver_base_pin pin_name
```

Default: `{}`

Data_type: string

Specifies the base pin that will be used for all timing and capacitance modeling for clock roots with a source pin at a blackbox output.

cts_blackbox_default_load_base_pin

```
cts_blackbox_default_load_base_pin pin_name
```

Default: `{}`

Data_type: string

Specifies the base pin that will be used for all timing and capacitance modeling for clock sinks at blackbox inputs. Any `cts_capacitance_override` attribute will be used in preference to this.

cts_buffer_cells

```
cts_buffer_cells buffer_cell_list
```

Default: `{}`

Data_type: string, read/write

Specifies the buffer cells for CTS. If none are specified CTS will choose buffers from the libraries. Cell names may be specified as a Tcl list of names, or as a Tcl list of patterns to be expanded to match names. If set explicitly, CTS will ignore any don't use settings for the cells specified. Different buffer cells may be specified for any combination of clock tree and power domain.

To use different buffers for each net type set the `cts_buffer_cells_top` and `cts_buffer_cells_leaf` attributes.

Some examples are provided below:

To specify buffer cells for all clock trees and all power domains:

```
set_db cts_buffer_cells {bufAX* bufBX*}
```

To specify buffer cells for a particular clock tree and all power domains:

```
set_db clock_tree:<clk> .cts_buffer_cells {bufX20 bufX18}
```

To specify buffer cells for a particular clock tree and power domain:

```
set_db clock_tree:<clk> .cts_buffer_cells -index {power_domain <pd>} {bufX12 bufX8}
```

Valid values: a list of library cell names, or a list of patterns to expand to library cell names

Note: This attribute is applicable to object types, `clock_tree` and `power_domain`.

cts_buffer_cells_leaf

```
cts_buffer_cells_leaf  buffer_cell_list
```

Default: `{}`

Data_type: string, read/write

Specifies the buffer cells available for CTS to use on leaf nets. If none are specified, CTS will use the same buffers as on trunk nets - as specified in the `cts_buffer_cells` attribute. Cell names may be specified as a Tcl list of names, or as a Tcl list of patterns to be expanded to match names. If set explicitly, CTS will ignore any don't use settings for the cells specified.

Different leaf buffer cells may be specified for any combination of clock tree and power domain.

Valid values: library cell names, or a list of patterns to expand to library cell names

Note: This attribute is applicable to object types, `clock_tree` and `power_domain`.

cts_buffer_cells_top

`cts_buffer_cells_top` *`buffer_cell_list`*

Default: `{}`

Data_type: string, read/write

Specifies the buffers cells available for CTS to use on top nets. If none are specified, CTS will use the same buffers as on trunk nets - as specified in the `cts_buffer_cells` attribute. Cell names may be specified as a Tcl list of names, or as a Tcl list of patterns to be expanded to match names. If set explicitly, CTS will ignore any don't use settings for the specified cells.

Different top buffer cells may be specified for any combination of clock tree and power domain.

Valid values: a list of library cell names, or a list of patterns to expand to library cell names

Note: This attribute is applicable to object types, `clock_tree` and `power_domain`.

cts_cell_density

`cts_cell_density <`*`value`*`>`

Default: 0.75

Valid Values: `0.01 to 1`

Data_type: string, read/write

Specifies the clock halo in the x direction. Any x direction clock halo defined by this attribute is proportional to the cell width. The constant of proportionality is defined by the attribute value. For example:

- If `cts_cell_density = 0.25` then the x direction clock halo equals 3 * cell width.
- If `cts_cell_density = 0.5` then the x direction clock halo equals cell width.
- If `cts_cell_density = 0.75` then the x direction clock halo equals cell width / 3.
- If `cts_cell_density = 1` then the x direction clock halo equals zero.

This attribute can specify the x direction clock halo for all clock trees via:

```
set_db cts_cell_density 0.8
```

This attribute can specify the x direction clock halo per-clock tree via:

```
set_db clock_tree:<clk> .cts_cell_density 0.9
```

The following attributes can be used to assign x direction clock halos within CTS:

- `cts_cell_halo_x`
- `cts_cell_density`
- `cts_cell_halo_sites`

Only one of these attributes is used to determine the clock halo in the x direction. The following rules determine which:

- If `cts_cell_halo_x` is set to a non-auto value, then this defines the x direction clock halo. The attributes `cts_cell_density` and `cts_cell_halo_sites` have no effect.
- If `cts_cell_halo_x` is set to auto and `cts_cell_density` is set to a non-auto value then `cts_cell_density` defines the clock halo in the x direction. The attribute `cts_cell_halo_sites` has no effect.
- If both `cts_cell_halo_x` and `cts_cell_density` are set to auto then `cts_cell_halo_sites` defines the clock halo in the x direction.

cts_cell_halo_mode

```
cts_cell_halo_mode {max | sum}
```

Default: `max`

Data_type: string, read/write

Specifies how clock halos are used to determine the minimum legal separation between a pair of clock instances. There are two possible modes 'max' and 'sum'. When set to 'max' the minimum legal separation is the larger of the two clock halos. When set to 'sum' the minimum legal separation is the sum of the two clock halos.

Valid values: `max sum`

See also:
. cts_cell_halo_x
. cts_cell_halo_y
. cts_cell_density
. cts_adjacent_rows_legal
. cts_cell_halo_sites
. cts_cell_halo_rows
. cts_effective_clock_halo_x
. cts_effective_clock_halo_y
. cts_effective_clock_halo_x_source
. cts_effective_clock_halo_y_source

cts_cell_halo_rows

```
cts_cell_halo_rows value
```

Default: `1`

Data_type: int, read/write

Specifies the clock halo in the y direction in rows for all clock cells.

The following attributes can be used to assign y direction clock halos within CCOpt:

- `cts_cell_halo_y`
- `cts_adjacent_rows_legal`
- `cts_cell_halo_rows`

Only one of these attributes is used to determine the clock halo in the y direction. The following rules determine which:

- If cts_cell_halo_y is set to a non-auto value, then this defines the y direction clock halo. The attributes cts_adjacent_rows_legal and cts_cell_halo_rows have no effect.
- If cts_cell_halo_y is set to auto and cts_adjacent_rows_legal is set to a non-auto value then cts_adjacent_rows_legal defines the clock halo in the y direction. The attribute cts_cell_halo_rows has no effect.
- If both cts_cell_halo_y and cts_adjacent_rows_legal are set to auto then cts_cell_halo_rows defines the clock halo in the y direction.

See also:
.cts_cell_halo_y
. cts_adjacent_rows_legal

cts_cell_halo_sites

```
cts_cell_halo_sites value
```

Default: `4`

Data_type: int, read/write

Specifies the clock halo in the x direction in sites for all clock cells.

The following attributes can be used to assign x direction clock halos within CCOpt:

- `cts_cell_halo_x`
- `cts_cell_density`
- `cts_cell_halo_sites`

Only one of these attributes is used to determine the clock halo in the x direction. The following rules determine which:

- If cts_cell_halo_x is set to a non-auto value, then this defines the x direction clock halo. The attributes cts_cell_density and cts_cell_halo_sites have no effect.
- If cts_cell_halo_x is set to auto and cts_cell_density is set to a non-auto value then cts_cell_density defines the clock halo in the x direction. The attribute cts_cell_halo_sites has no effect.
- If both cts_cell_halo_x and cts_cell_density are set to auto then cts_cell_halo_sites defines the clock halo in the x direction.

See also:
. `cts_cell_halo_x`
. `cts_cell_density`

cts_cell_halo_x

`cts_cell_halo_x` *`value`*

Default: `auto`

Data_type: string, read/write

Specifies the clock halo distance in the x direction. The default value of this attribute is auto.
The following attributes can be used to assign x direction clock halos within CTS:

- `cts_cell_halo_x`
- `cts_cell_density`
- `cts_cell_h`alo_sites

Only one of these attributes is used to determine the clock halo in the x direction. The following rules determine which:

- If `cts_cell_halo_x` is set to a non-auto value, then this defines the x direction clock halo. The attributes `cts_cell_density` and `cts_cell_halo_sites` have no effect.
- If `cts_cell_halo_x` is set to auto and cts_cell_density is set to a non-auto value then cts_cell_density defines the clock halo in the x direction. The attribute `cts_cell_halo_sites` has no effect.
- If both `cts_cell_halo_x` and `cts_cell_density` are set to auto then `cts_cell_halo_sites` defines the clock halo in the x direction.

See also:
.`cts_cell_density`
.`cts_cell_halo_sites`

cts_cell_halo_y

`cts_cell_halo_y` *`value`*

Default: `auto`

Data_type: string, read/write

Specifies the clock halo distance in the y direction. The default value of this attribute is auto.
The following attributes can be used to assign y direction clock halos within CTS:

- `cts_cell_halo_y`
- `cts_adjacent_rows_legal`
- `cts_cell_halo_rows`

Only one of these attributes is used to determine the clock halo in the y direction. The following rules determine which:

- If cts_cell_halo_y is set to a non-auto value, then this defines the y direction clock halo. The attributes cts_adjacent_rows_legal and cts_cell_halo_rows have no effect.
- If cts_cell_halo_y is set to auto and cts_adjacent_rows_legal is set to a non-auto value then cts_adjacent_rows_legal defines the clock halo in the y direction. The attribute cts_cell_halo_rows has no effect.
- If both cts_cell_halo_y and cts_adjacent_rows_legal are set to auto then cts_cell_halo_rows defines the clock halo in the y direction.

See also:
. `cts_adjacent_rows_legal`
. `cts_cell_halo_rows`

cts_clock_gating_cells

`cts_clock_gating_cells` *`cell_list`*

Default: `{}`

Data_type: string, read/write

Specifies the clock gates for CTS. If none are specified CTS will choose clock gates from the libraries. Cell names may be specified as a Tcl list of names, or as a Tcl list of patterns to be expanded to match names. If set explicitly, CTS will ignore any do not use settings for the cells specified.

A list of lists can be used to specify separate clock gate sub-lists. All CTS transforms will resize clock gates within the clock gate sub-list they are initially part of. If an incoming clock gate is not part of any clock gate sub-list, it will be resized to the first clock gate sub-list.

Different clock gates may be specified for any combination of clock tree and power domain.

Some examples are provided below:

To specify clock gates for all clock trees and all power domains:

```
set_db cts_clock_gating_cells {cgAX* cgBX*}
```

To specify clock gates for a particular clock tree and all power domains:

```
set_db clock_tree:<clk> .cts_clock_gating_cells {cgX20 cgX18}
```

To specify clock gates for a particular clock tree and power domain:

```
set_db clock_tree:<clk> .cts_clock_gating_cells -index {power_domain <pd>} {cgX12 cgX8}
```

To specify separate clock gate sub-lists:

```
set_db .cts_clock_gating_cells {{cgX12A1 cgX8A1} {cgX12A2 cgX8A2}}
```

Valid values: `a list of library cell names, or a list of patterns to expand to library cell names, a list of lists to specify separate clock gate sub-lists`

cts_clock_gate_movement_limit

`cts_clock_gate_movement_limit` *`value`*

Default: `10`

Data_type: string, read/write

Each clock gate is restricted to a Manhattan ball centered on its original location with the CTS flow. The radius of the ball is a multiple of the clock gate height. This attribute controls the default value of that multiple.

cts_clock_source_cells

`cts_clock_source_cells` *`list_of_cells`*

Default: `{}`

Data_type: string, read/write

Specifies the cells available for CTS to size clock sources if the attribute, `cts_size_clock_sources` is set to `true`. If none are specified CTS will choose cells from the libraries. Cell names may be specified as a Tcl list of names, or as a Tcl list of patterns to be expanded to match names.

If set explicitly, CTS will ignore any don't use settings for the specified cells.

Different cells may be specified for clock trees or power domains. Only clock sources that are buffers, inverters, logic, and clock gating cells with a single output can be resized.

Valid values: `a list of library cell names, or a list of patterns to expand to library cell names`

Optional applicable arguments: "`-clock_tree <`*`name`*`>`" and "`-power_domain <`*`name`*`>`".

cts_clock_tree_source_group_clock_trees

`cts_clock_tree_source_group_clock_trees` *`clock_tree_list`*

Default: `{}`

Data_type: string, read/write

A list of the clock trees relevant to this source group.

Valid values: `list clock_trees`

cts_clock_tree_source_input_max_transition_time

`cts_clock_tree_source_input_max_transition_time {double | 0}`

Default: `0`

Data_type: double, read/write

Specifies the slew that will be assumed at the input of the root driver.

Valid values: `double | 0`

cts_cloning_inst_name_suffix

`cts_cloning_inst_name_suffix` *`suffix_name`*

Default: `clone`

Data_type: string

If set, the suffix will be used generally for all insts cloned by CTS, else the suffix will default to "clone".

cts_cloning_inst_name_suffix_source_group_assignment

`cts_cloning_inst_name_suffix_source_group_assignment` *`suffix_name`*

Default: `{}`

Data_type: string, read/write

Specifically controls the suffix used for multi tap cloning. If this is set to a non-empty string, that string will be used as a suffix when naming cloned instances created by multi-tap cloning. Otherwise, the name suffix will be controlled by the `cts_cloning_inst_name_suffix` attribute.

cts_delay_cells

`cts_delay_cells` *`cell_list`*

Default: `{}`

Data_type: string, read/write

Specifies the delay cells available for CTS. If none are specified CTS will not use delay cells. Setting this attribute to the string 'auto' means that CTS will choose delay cells from the libraries to use. Cell names may be specified as a Tcl list of names, or as a Tcl list of patterns to be expanded to match names.

If set explicitly, CTS will ignore any don't use settings for the cells specified. Different delay cells may be specified for any combination of clock tree and power domain, or by omitting those arguments a global setting can be applied.

Some examples are provided below:

To specify delay cells for all clock trees and power domains:

```
set_db cts_delay_cells {delayAX* delayBX*}
```

To specify delay cells for a particular clock tree and all power domains:

```
set_db clock_tree:<clk> .cts_delay_cells {delayX1 delayX2}
```

To specify delay cells for a particular clock tree and power domain:

```
set_db clock_tree:<clk> .cts_delay_cells -index {power_domain <pd>} {delayX2 delayX3}
```

Valid values: `a list of library cell names, or a list of patterns to expand to library cell names, or the string 'auto'`

cts_detailed_cell_warnings

`cts_detailed_cell_warnings {true | false}`

Default: `false`

Data_type: bool, read/write

If set to `true`, CTS outputs detailed cell warning diagnostics when it encounters issues with library cell selection, power domains and/or signal levels.

Valid values: `true, false`

cts_exit_if_no_placeable_area

`cts_exit_if_no_placeable_area {true | false}`

Default: `true`

Data_type: bool, read/write

End CTS during initialization if there is insufficient placeable area.

cts_exit_if_skew_target_over_constrained

`cts_exit_if_skew_target_over_constrained {true | false}`

Default: `true`

Data_type: bool, read/write

End CTS during initialization if the skew target is too low.

Valid values: `true, false`

cts_exit_if_stage_delay_sigma_target_over_constrained

`cts_exit_if_stage_delay_sigma_target_over_constrained {true | false}`

Default: `true`

Data_type: bool, read/write

If set, allows any max stage delay sigma target to be set. If the target is too low, it is likely to lead to runtime problems.

Valid values: `true, false`

cts_fix_clock_sinks

`cts_fix_clock_sinks {true | false | soft}`

Default: `false`

Data_type: string, read/write

If set to `true`, we will DEF lock clock tree sinks after routing in addition to any clock node locking (fixed).

If set to `soft`, we will DEF lock clock tree sinks after routing in addition to any clock node soft_locking (softFixed).

Valid values: `true false soft`

cts_flexible_htree_enable_metrics_reporting

`cts_flexible_htree_enable_metrics_reporting {true | false | post_route}`

Default: `post_route`

Data_type: string, read/write

Controls the reporting of metrics for flexible H-trees. The following values are allowed:

- `true`: Enables all reporting.
- `false`: Disables all reporting.
- `post_route`: Enables only the report at the end of flexible H-Tree synthesis.

Valid values: `true false post_route`

cts_flexible_htree_partition_boundary_polarity

`cts_flexible_htree_partition_boundary_polarity {non_inverting | inverting | ignore}`

Default: `non_inverting`

Data_type: string, read/write

Specifies the polarity of the clock signal with regard to the source of the flexible H-tree when entering partitions. The following values are allowed:

- `non_inverting`: When specified, allows only non-inverting polarity at partition entry points.
- `inverting`: When specified, allows only inverting polarity at partition entry points.
- `ignore`: When specified, allows inverting and non-inverting polarity at partition entry points.

Valid values: `non_inverting inverting ignore`

cts_flexible_htree_placement_legalization_effort

```
cts_flexible_htree_placement_legalization_effort {high | low}
```

Default: `low`

Data_type: string, read/write

The legalization effort for finding placement unblocked points on the synthesis grid for flexible H-trees. High placement legalization effort can avoid having to relax placement constraints when implementing H-trees but may lead to increased runtime.

Valid values: `high low`

cts_force_target_max_transition_time

```
cts_force_target_max_transition_time {true | false}
```

Default: `false`

Data_type: bool, read/write

Specifies that CTS should allow any transition target to be used, even if it is likely to lead to runtime problems. By default, CTS computes a minimum transition target which should not lead to runtime problems, and does not allow any transition target to be set lower than this minimum. This attribute overrides that check, and allows any transition target to be used.

Valid values: `true, false`

cts_ignore_problematic_skew_as_result_of_dont_touch_nets

```
cts_ignore_problematic_skew_as_result_of_dont_touch_nets {true | false}
```

Default: `false`

Data_type: bool, read/write

If set to true, sinks directly connected to nets that are causing unfixable skew problems will be ignored for skew balancing.

Valid values: `true, false`

cts_inst_name_prefix

```
cts_inst_name_prefix prefix_name
```

Default: `CTS`

Data_type: string, read/write

The name prefix of instances/nets created by CTS. The default value is "CTS". The default names of nets are CTS, CTS_1, CTS_2... ...The default names of instances are CTS_*.

Valid values: `string`

cts_inverter_cells

```
cts_inverter_cells inverter_cell_list
```

Default: `{}`

Data_type: string, read/write

Specifies the inverter cells available for CTS. If none are specified CTS will choose inverters from the libraries. Cell names may be specified as a Tcl list of names, or as a Tcl list of patterns to be expanded to match names. If set explicitly, CTS will ignore any don't use settings for the cells specified. Different inverter cells may be specified for any combination of clock tree and power domain.

To use different inverters for each net type set the `cts_inverter_cells_top` and `cts_inverter_cells_leaf` attributes.

Some examples are provided below:

To specify inverter cells for all clock trees and all power domains:
```
set_db cts_inverter_cells {invAX* invBX*}
```

To specify inverter cells for a particular clock tree and all power domains:
```
set_db clock_tree:<clk> .cts_inverter_cells {invX20 invX18}
```

To specify inverter cells for a particular clock tree and power domain:
```
set_db clock_tree:<clk> .cts_inverter_cells -index {power_domain <pd>} {invX12 invX8}
```

Valid values: a list of library cell names, or a list of patterns to expand to library cell names

cts_inverter_cells_leaf

`cts_inverter_cells_leaf` *inverter_cell_list*

Default: `{}`

Data_type: string, read/write

Specifies the inverter cells available for CTS to use on leaf nets. If none are specified, CTS will use the same inverters as on trunk nets - as specified in the `cts_inverter_cells` attribute. Cell names may be specified as a Tcl list of names, or as a Tcl list of patterns to be expanded to match names. If set explicitly, CTS will ignore any don't use settings for the cells specified.

Different leaf inverter cells may be specified for any combination of clock tree and power domain.

Valid values: a list of library cell names, or a list of patterns to expand to library cell names

Note: This attribute is applicable to object types, `clock_tree` and `power_domain`.

cts_inverter_cells_top

`cts_inverter_cells_top` *cell_list*

Default: `{}`

Data_type: string, read/write

Specifies the inverter cells available for CTS to use on top nets. If none are specified, CTS will use the same inverters as on trunk nets - as specified in the `cts_inverter_cells` attribute. Cell names may be specified as a Tcl list of names, or as a Tcl list of patterns to be expanded to match names.If set explicitly, CTS will ignore any don't use settings for the cells specified.

Different top inverter cells may be specified for any combination of clock tree and power domain.

Valid values: a list of library cell names, or a list of patterns to expand to library cell names

Note: This attribute is applicable to object types, `clock_tree` and `power_domain`.

cts_load_capacitance_cells

`cts_load_capacitance_cells` *list_of_cellnames*

Default: `{}`

Data_type: string, read/write

Specifies the load capacitance cells available for CTS. CTS will use cells from this collection for load capacitance optimizations. Cell names may be specified as a Tcl list of names, or as a Tcl list of patterns to be expanded to match names.

Note: This attribute is applicable to object types, `clock_tree` and `power_domain`.

cts_logic_cells

`cts_logic_cells` *clock_logic_cell_list*

Default: `{}`

Data_type: string, read/write

Specifies the clock logics for CTS. If none are specified CTS will choose clock logics from the libraries. Cell names may be specified as a Tcl list of names, or as a Tcl list of patterns to be expanded to match names. If set explicitly, CTS will ignore any don't use settings for the cells specified. Different logic cells may be specified for any combination of clock tree and power domain.

Some examples are provided below:

To specify logic cells for all clock trees and all power domains:

```
set_db cts_logic_cells {and* mux*}
```

To specify logic cells for a particular clock tree and all power domains:

```
set_db clock_tree:<clk> .cts_logic_cells {andX20 andX18}
```

To specify logic cells for a particular clock tree and power domain:

```
set_db clock_tree:<clk> .cts_logic_cells -index {power_domain <pd>} {andX12 andX8}
```

Valid values: `a list of library cell names, or a list of patterns to expand to library cell names`

cts_manage_power_intent_violations

```
cts_manage_power_intent_violations {true | false}
```

Default: `true`

Data_type: bool, read/write

If this attribute is set, the CTS algorithm will work around power management illegalities in the clock tree, as opposed to failing with an error when it encounters them. This allows the clock tree to be synthesized, but any power
management illegalities will remain in the exported design.

Valid values: `true, false`

cts_max_fanout

```
cts_max_fanout number
```

Default: `100`

Data_type: int, read/write

The maximum fanout at any point in the clock tree.

Valid values: integer ranged between `2` and `1000` inclusive

Note: This attribute is applicable to object type, `base_pin`. For detailed help of the attribute in the tool, type `help <obj_type> <attribute_name>`. For example: `help base_pin cts_max_fanout`

cts_max_instance_height

```
cts_max_instance_height value
```

Default: `3`

Data_type: double, read/write

When selecting cells for use, CTS will filter the selection to remove all buffers, inverters and clock gates taller than this many row heights. This helps quickly eliminate very large cells that are unlikely to be useful.

cts_max_source_to_sink_net_length

```
cts_max_source_to_sink_net_length {auto | double}
```

Default: `auto`

Data_type: string, read/write

The maximum routing length in microns between driving source pin and driven sink pin on each net that clock tree synthesis should observe. This constraint can be applied to either a pin, a clock tree, or a net type. By default (if this attribute is not set) no explicit clock tree net length constraint is enforced. However, other clock tree constraints such as maximum slew (transition) and maximum capacitance will indirectly limit the maximum net length.

Valid values: `auto | double`

cts_max_source_to_sink_net_length_leaf

```
cts_max_source_to_sink_net_length_leaf {auto | double}
```

Default: `auto`

Data_type: string, read/write

The maximum routing length in microns between driving source pin and driven sink pin on each net that clock tree synthesis should observe. This constraint can be applied to either a pin, a clock tree, or a net type. By default (if this attribute is not set) no explicit clock tree net length constraint is enforced. However, other clock tree constraints such as maximum slew (transition) and maximum capacitance will indirectly limit the maximum net length. This attribute only applies to leaf net type.

Valid values: `auto | double`

cts_max_source_to_sink_net_length_top

```
cts_max_source_to_sink_net_length_top {auto | double}
```

Default: `auto`

Data_type: string, read/write

The maximum routing length in microns between driving source pin and driven sink pin on each net that clock tree synthesis should observe. This constraint can be applied to either a pin, a clock tree, or a net type. By default (if this attribute is not set) no explicit clock tree net length constraint is enforced. However, other clock tree constraints such as maximum slew (transition) and maximum capacitance will indirectly limit the maximum net length. This attribute only applies to top net type.

Valid values: `auto | double`

cts_max_source_to_sink_net_length_trunk

```
cts_max_source_to_sink_net_length_trunk {auto | double}
```

Default: `auto`

Data_type: string, read/write

The maximum routing length in microns between driving source pin and driven sink pin on each net that clock tree synthesis should observe. This constraint can be applied to either a pin, a clock tree, or a net type. By default (if this attribute is not set) no explicit clock tree net length constraint is enforced. However, other clock tree constraints such as maximum slew (transition) and maximum capacitance will indirectly limit the maximum net length. This attribute only applies to trunk net type.

Valid values: `auto | double`

cts_merge_clock_gates

```
cts_merge_clock_gates {true | false}
```

Default: `false`

Data_type: bool, read/write

If set to `true`, clock gate merging is enabled. If this is false, merging of all clock gates is disabled, including clock gates which may have been cloned by CTS.

This attribute also impacts the early clock flow.

Note that this attribute only impacts '`ccopt_design -cts`'. See also attribute `cts_merge_clock_logic`.

Valid values: `true, false`

cts_merge_clock_logic

```
cts_merge_clock_logic {true | false}
```

Default: `false`

Data_type: bool, read/write

If set to `true`, clock logic merging is enabled. If this is `false`, merging of all clock logics is disabled, including clock logics which may have been cloned by CTS.

Note that this attribute only impacts '`ccopt_design -cts`'. See also attribute `cts_merge_clock_gates`.

Valid values: `true, false`

cts_mixed_fanout_net_type

```
cts_mixed_fanout_net_type  {leaf | trunk}
```

Default: `leaf`

Data_type: string, read/write

Controls how CTS considers nets that have fanout consisting partially but not entirely of sinks. By default, having any sinks, for example DFFs, will make the net be considered leaf, but when set to trunk, a net has to drive only sinks to be considered as leaf. For example, a net driving a clock gate and a DFF would no longer count as a leaf net.

Valid values: `leaf trunk`

cts_move_clock_gates

```
cts_move_clock_gates {true | false}
```

Default: `true`

Data_type: bool, read/write

If this attribute is set, the CTS algorithm will move clock gates that appear in the clock tree. 'Logic' does not include clock gates, buffers, and inverters in a clock tree, which are always moved unless they are locked, or clock generators that are above the root of the clock tree. Usually, this will affect multiplexers used for selecting one of a number of clocks, or for switching between a test clock and the main clock. Setting this attribute may cause the clock tree to have a lower insertion delay, but might break datapath timing. During optimization, this isn't a problem, because the timing will be automatically recovered during the optimization process.

Note: The cts_move_clock_gates attribute indirectly controls multi-tap CTS because the logic is that if no integrated clock gates (ICGs) can be moved then multi-tap cloning is also disabled. So, it is recommend to not use `cts_move_clock_gates false` for multi-tap CTS.

cts_move_logic

```
cts_move_logic {true | false}
```

Default: `true`

Data_type: bool, read/write

If this attribute is set, the CTS algorithm will move logic that appears in the clock tree. "Logic" does not include clock gates, buffers, and inverters in a clock tree, which are always moved unless they are locked, or clock generators that are above the root of the clock tree. Usually, this will affect multiplexers used for selecting one of a number of clocks, or for switching between a test clock and the main clock. Setting this attribute may cause the clock tree to have a lower insertion delay, but might break datapath timing. During optimization, this isn't a problem, because the timing will be automatically recovered during the optimization process.

Valid values: `true, false`

cts_net_name_prefix

```
cts_net_name_prefix string
```

Default: `CTS`

Data_type: string, read/write

The name prefix of instances/nets created by CTS. The default value is "CTS". The default names of nets are CTS, CTS_1, CTS_2... ...The default names of instances are CTS_*.

Valid values: `string`

cts_override_vias

```
cts_override_vias via_name_list
```

Default: `{}`

Data_type: string, read/write

When specified, this attribute defines the vias to be used in RC extraction from routing estimates. The listed vias will be used in place of those configured on the routing rules for the clock network. The order of the list is irrelevant. The vias may also be overridden for each non-default rule (NDR). In this case, the name of the NDR is given as the first element in the list with subsequent entries being the via names. Multiple NDRs can be specified, for example, `{via1d {NDR1 via2d} {NDR2 via2d}}` will override `via1d` in the default rule, and both `via1d` and `via2d` in `NDR1` and `NDR2`.

Valid values: `list via_call`

cts_pin_insertion_delay_histogram_bin_size

`pin_insertion_delay_histogram_bin_size {auto | time value in library units}`

Default: `auto`

Data_type: string, read/write

This attribute specifies the preferred size of the histogram bins that should be used when reporting the distribution of pin insertion delays. The bin size is specified in time units and must be greater than zero. A value of auto (the default) leaves it to the tool to calculate a bin size.

This attribute affects pin insertion delay histograms that are logged during various flow commands. It also determines the preferred histogram bin size that is used by the `report_pin_insertion_delays` command when an explicit `-bin_size` argument has not been specified.

Valid values: `auto, time value in library units`

cts_post_route_enable

`cts_post_route_enable {true | false}`

Default: `true`

Data_type: bool, read/write

Enable post-conditioning optimization after clock nets are routed.

Valid values: `true false`

cts_post_route_repair_drv

`cts_post_route_repair_drv {true | false}`

Default: `true`

Data_type: bool, read/write

If set to `false`, post-conditioning will skip its DRV fixing step.

Valid values: `true false`

cts_post_route_enable_routing_eco

`cts_post_route_enable_routing_eco {true | false}`

Default: `false`

Data_type: bool, read/write

If set to `false`, post-conditioning will skip its ECO-routing step.

Valid values: `true false`

cts_post_route_repair_skew_by_buffering

`cts_post_route_repair_skew_by_buffering {true | false}`

Default: `false`

Data_type: bool, read/write

If set to `true`, post-conditioning will attempt skew-fixing using rebuffering.

Valid values: `true false`

cts_primary_delay_corner

`cts_primary_delay_corner corner_name`

Default: `{}`

Data_type: string, read/write

This specifies the delay corner in which clock tree balancing applies the slew and insertion delay targets. If more than one timing corner is defined, this must be set before running CTS. By default, this is set to the first defined delay corner.

Valid values: `corner name`, or `empty`

cts_primary_reporting_skew_groups

`cts_primary_reporting_skew_groups` *`list of skew groups`* `| auto | none`

Default: `auto`

Data_type: string, read/write

Specifies the primary skew groups used for reporting. By default, the value is specified as auto that automatically takes the skew group with maximum number of sinks as the primary reporting skew group. For invalid values, the default (auto) will be considered.

Valid values: a `list of existing skew group names, auto` or `none`

cts_primary_reporting_skew_groups_log_min_max_sinks

`cts_primary_reporting_skew_groups_log_min_max_sinks on | off | logv`

Default: `logv`

Data_type: string, read/write

If set to `on`, the sinks with the shortest and longest paths in each primary reporting skew group will be logged. If set to `logv`, they will be logged only to the logv file.

Valid values: `on, off, logv`

cts_repair_congestion

`cts_repair_congestion {true | false}`

Default: `true`

Data_type: bool, read/write

If set, place_fix_congestion is called during CTS. This can reduce congestion and ease later routing issues.

Valid values: `true false`

cts_repair_drv_by_buffering

`cts_repair_drv_by_buffering {true | false}`

Default: `false`

Data_type: bool, read/write

If set to `true`, post-conditioning will fix DRVs by adding buffers.

Valid values: `true false`

cts_report_skew_groups_only_with_targets

`cts_report_skew_groups_only_with_targets {true | false}`

Default: `false`

Data_type: bool, read/write

The skew groups report (run using the `report_skew_groups` command) displays insertion delay, skew, and min/max path information for different combinations of skew group, timing corner, and early/late path. Set the `cts_report_skew_groups_only_with_targets` attribute to `false` (the default) to report on all skew group/timing corner/path combinations regardless of whether a skew target has been set. Set the `cts_report_skew_groups_only_with_targets attribute` to `true` to report only on skew group/timing corner/path combinations where a skew target has been set (either explicitly or using the 'auto' setting).

cts_route_clock_tree_nets

```
cts_route_clock_tree_nets {true | false}
```

Default: `true`

Data_type: bool, read/write

Specifies that detailed routing be performed during the final implementation clock routing phase.

cts_route_type_leaf

```
cts_route_type_leaf {route_type | default}
```

Default: `default`

Data_type: string, read/write

Specifies the route type. Setting this attribute binds an existing user-defined route_type to one or more types of clock tree nets. Binding a route_type to a type of clock tree nets means that all nets of that type (including the nets created by CTS) will be routed according to the specification of that route_type. In the most common usage, the route_type is bound to one of the three types of clock tree nets (top, trunk, or leaf) with the optional -net_type argument. Omitting the -net_type argument causes the route_type to be bound to all three types of clock tree nets. The optional -clock_tree <pattern> argument limits the binding to the clock trees whose name matches <pattern>. Omitting the -clock_tree argument causes the binding to apply to all clock trees.

For a route_type to be used in CTS, it must be bound to at least one net type. If net type is not bound to any route_type, a default route_type will be created for that net type at the start of CTS.

Valid values: names of route_types created with create_route_type

cts_route_type_top

```
cts_route_type_top {route_type | default}
```

Default: `default`

Data_type: string, read/write

Specifies the route type. Setting this attribute binds an existing user-defined route_type to one or more types of clock tree nets. Binding a route_type to a type of clock tree nets means that all nets of that type (including the nets created by CTS) will be routed according to the specification of that route_type. In the most common usage, the route_type is bound to one of the three types of clock tree nets (top, trunk, or leaf) with the optional -net_type argument. Omitting the -net_type argument causes the route_type to be bound to all three types of clock tree nets. The optional `-clock_tree <pattern>` argument limits the binding to the clock trees whose name matches <pattern>. Omitting the -clock_tree argument causes the binding to apply to all clock trees.

For a route_type to be used in CTS, it must be bound to at least one net type. If net type is not bound to any route_type, a default route_type will be created for that net type at the start of CTS.

Valid values: names of route_types created with create_route_type

cts_route_type_trunk

```
cts_route_type_trunk {route_type | default}
```

Default: `default`

Data_type: string, read/write

Specifies the route type. Setting this attribute binds an existing user-defined route_type to one or more types of clock tree nets. Binding a route_type to a type of clock tree nets means that all nets of that type (including the nets created by CTS) will be routed according to the specification of that route_type. In the most common usage, the route_type is bound to one of the three types of clock tree nets (top, trunk, or leaf) with the optional -net_type argument. Omitting the -net_type argument causes the route_type to be bound to all three types of clock tree nets. The optional -clock_tree <pattern> argument limits the binding to the clock trees whose name matches <pattern>. Omitting the -clock_tree argument causes the binding to apply to all clock trees.

For a route_type to be used in CTS, it must be bound to at least one net type. If net type is not bound to any route_type, a default route_type will be created for that net type at the start of CTS.

Valid values: names of route_types created with create_route_type

cts_routing_preferred_layer_effort

```
cts_routing_preferred_layer_effort {standard | high | auto}
```

Default: `standard`

Data_type: string, read/write

Controls preferred layer effort for clock routing. The following settings may be specified:

- `standard` (default) - No change made to current early global route settings.
- `high` - enables a package of early global route settings to increase layer adherence, possibly at the expense of wire length and via count.
- `auto` - maps to `high` effort only if the effective routing effort level for the vast majority of nets of `net_type trunk`, for all clock trees resolves to `high`. Otherwise, the `auto` setting maps to `standard`.

cts_size_clock_gates

```
cts_size_clock_gates {true | false}
```

Default: `true`

Data_type: bool, read/write

When set to true (the default), the CTS algorithm sizes clock gates that appear in the clock tree. 'Logic' does not include clock gates, buffers, and inverters in a clock tree, which are always sized unless they are locked, or clock generators that are above the root of the clock tree. Usually, this affects multiplexers used for selecting one of a number of clocks, or for switching between a test clock and the main clock. Setting this attribute may cause the clock tree to have a lower insertion delay, but might change the cell types of logic gates in the clock tree, which in turn may require them to be moved slightly to find a legal location for the new cell.

cts_size_clock_sources

```
cts_size_clock_sources {true | false}
```

Default: `false`

Data_type: bool, read/write

When set to true, CTS will try to size the clock source. Only clock sources that are buffers, inverters, logic and clock gating cells with a single output will be sized. The cells available for CTS to size clock sources can be set specified using the attribute, `cts_clock_source_cells`.

cts_size_logic

```
cts_size_logic {true | false}
```

Default: `true`

Data_type: bool, read/write

When set to `true` (the default), the CTS algorithm sizes logic that appears in the clock tree. "Logic" does not include clock gates, buffers, and inverters in a clock tree, which are always sized unless they are locked, or clock generators that are above the root of the clock tree. Usually, this affects multiplexers used for selecting one of a number of clocks, or for switching between a test clock and the main clock. Setting this attribute may cause the clock tree to have a lower insertion delay, but might change the cell types of logic gates in the clock tree, which in turn may require them to be moved slightly to find a legal location for the new cell.

Valid values: `true, false`

cts_skew_group_report_columns

```
cts_skew_group_report_columns list_of_columns
```

Default: `{name lib_cell event increment time slew capacitance location distance fanout status}`

Data_type: string, read/write

A Tcl list of columns to include in skew group reports produced by the skew group report. You can use this attribute to specify the columns you would like to include in the skew group report, and the order in which the columns should appear. Most of the legal values are straightforward. However, the set of legal values of the following form deserve further explanation:

```
summaryType_summaryLocation[_event]
```

These values let you report the delay value for other paths that go through the pin.

`summaryType` is one of: max (show the longest delay), min (show the shortest delay), or skew (show the skew, that is the difference between the longest and the shortest delay).

`summaryLocation` is one of: above (show the delay/skew above this pin), below (show the delay/skew below this pin), or through (show the delay/skew for paths through this pin).

`event` is one of: rise (show the delay/skew for the rise event at this pin), fall (show the delay/skew for the fall event at this pin), or both (show the delay/skew for both events at this pin).

Valid values:
`capacitance`
`distance`
`event`
`fanout`
`increment`
`lib_cell`
`location`
`name`
`slew`
`status`
`time`

cts_skew_group_report_histogram_bin_size

```
cts_skew_group_report_histogram_bin_size {auto | string}
```

Default: `auto`

Data_type: string, read/write

When set to a numeric value, that numeric value will be used as the histogram range size (in library units). For example, if the library time units are set to 1 nanosecond, a value of 0.010 for `cts_skew_group_report_histogram_bin_size` will result in histogram ranges of 10 picoseconds.

When set to auto, the size of the histogram ranges are dependent on the skew targets that are set. If a skew target is set for a given half corner and skew group combination, then the histogram range size will be 10% of the skew target for that half corner and skew target combination. If no skew target is set for a half corner and skew group combination but a skew target is set for the primary half corner and skew group combination, then the histogram range size will be 10% of the skew target for the primary half corner and skew group combination.

In the event that no skew targets are set and `cts_skew_group_report_histogram_bin_size` is set to auto, a default value of 10 picoseconds will be used for the histogram range size.

Valid values: `auto | string`

cts_spec_config_base_pin_trace_through_to

```
cts_spec_config_base_pin_trace_through_to base_pin
```

Default: `{}`

Data_type: string, read/write

Clock tree definition will, by default, not continue through certain types of cell arc (for instance, the clock to Q arc in a DFF). This attribute allows you to override the default behavior by specifying the output base_pin to which the
clock tree should propagate, when it arrives at an instance of the given input library pin. The specified output base_pin must be another pin on the same library cell as the given input pin. The configured value is applicable to all instances of the
library cell in the netlist.

The output pin may be specified either by its fully qualified name (i.e. inclusive of the base_cell name), or else simply by its local (cell-relative) name.

Note: if both trace_through_to and library_trace_through_to are applicable at a given netlist instance pin, the trace_through_to value takes precedence.

Valid values: `base_pin`

cts_spec_config_create_clock_tree_source_groups

```
cts_spec_config_create_clock_tree_source_groups {true | false}
```

Default: `false`

Data_type: bool, read/write

Causes `create_clock_tree_spec` to set up a clock tree source group for SDC clocks with multiple source pins. If this attribute is set to active, `create_clock_tree_spec` defines one clock tree for each source pin
and then uses the `create_clock_tree_source_group` command to collect those clock trees together, so that CTS can distribute sinks between the clock trees. If this attribute is set to `inactive` or `true`, the source groups are also created during `create_clock_tree_spec`, but they will be defined as inactive.

Only active source groups will take part in sink allocation.

By default, it is set to `false`, which means that no source groups are extracted from SDC multiple pin clocks.

cts_spec_config_create_generator_skew_groups

```
cts_spec_config_create_generator_skew_group {true | false}
```

Default: `true`

Data_type: bool, read/write

This attribute will cause the `create_clock_tree_spec` command to create skew groups for sequential generators and their adjacent registers. Such skew groups will be specified with the same highest rank so that they can be balanced from the other normal skew groups that share some sinks of them. The adjacent registers of a generator are registers that have a datapath timing path to talk with the generator directly. When this attribute is set to `1`, one skew group will be created per sequential generator instance, master clock and generated clock tree triple. The resulting skew groups will by default be named in the pattern:

```
_clock_gen_<master_clock_name>_<generator_local_name><_optional_number>/<constraint_mode_name>.
```

For example, for a pair of generators, with the same local name "`reg_clkgen`", CTS creates generated clock trees from the same master clock named "`fclk`" in a constraint mode named "`func`" the skew groups emitted into the clock tree specification file would be named:

```
_clock_gen_fclk_reg_clkgen_1/func
_clock_gen_fclk_reg_clkgen_2/func
```

The prefix for the names of such skew groups is controlled by the cts_spec_config_create_generator_skew_group_name_prefix CTS attribute and defaults to "`_clock_gen`" (the start underscore is used to group
such skew groups at the end of any skew group listing ordered by name).

cts_spec_config_create_generator_skew_groups_name_prefix

```
cts_spec_config_create_generator_skew_groups_name_prefix prefix_name
```

Default: `_clock_gen`

Data_type: string, read/write

This attribute controls the skew group name prefix used for skew groups generated to balance generator flops with their adjacent flops. Default is "`_clock_gen`". The default has a start underscore at the beginning to cause listings of skew groups ordered by name to collect such skew groups together at the end of a list.

cts_spec_config_create_reporting_only_skew_groups

```
cts_spec_config_create_reporting_only_skew_groups {true | false}
```

Default: `true`

Data_type: bool, read/write

Controls whether spec creation will synthesize reporting-only skew groups.

A reporting-only skew group is a skew group whose `cts_skew_group_constrains` attribute is set to `none`. Such a skew group imposes no clock balancing constraint and will not be considered by CTS.

The typical example is a generated clock that is synchronous to its master clock. The relationship between the two clocks implies that the sinks of both clocks should be balanced together. To model this, a clock spec created for such a design will have the skew group corresponding to the master clock span the domain of both the master clock and also the generated clock. The skew group of the generated clock is then a subset of the master clock skew group: as such it is a redundant constraint, and may be safely omitted. If it is synthesized, such a skew group will be created as a reporting-only skew group.

When this attribute is set to `false`, reporting-only skew groups are completely omitted from the generated spec. When set to `true`, reporting-only skew groups are included in the generated spec, but are marked with the `cts_skew_group_constrains` attribute set to `none`. Either way, they impose no balancing constraint.

Valid values: `true false`

cts_spec_config_trace_multi_output_cells_with_single_clock_output

```
cts_spec_config_trace_multi_output_cells_with_single_clock_output {true | false | combinational}
```

Default: `combinational`

Data_type: string, read/write

This attribute controls whether and how `create_clock_tree_spec` will extract through cells that have multiple outputs but where only one of those outputs has a clock emanating from it; call these types of instance
multi-output single-clock instances.

If the feature is disabled, spec creation will not trace through multi-output single-clock instances. The resulting clock spec will specify a generated clock tree on

the clock-carrying outputs of every multi-output instace. Clock-carrying inputs that are connected to each clock output by way of timing arcs will be generator inputs to the corresponding generated clock tree. The presence of these generated clock trees may impact CTS operations on the instance such as sizing, moving, and cloning.

If the feature is enabled, the resulting clock spec will include cts_spec_config_trace_through_to settings for multi-output single-clock instances. These settings will allow the instance to be treated as a normal clock instance, most commonly as a clock logic. In this case there will be no need for the generated clock tree on the clock-carrying output. The treatment of instances with multiple clock-carrying outputs is unchanged: on such an instance each clock-carrying output will take a generated clock trees, as before.

The feature may be enabled conditionally, such that it is only applied to multi-output single-clock instance which are combinational. In this case multi-output single-clock instances which are sequential will continue to take a generated clock tree on the clock output.

The possible values for this attribute are as follows:

- `false` - The feature is completely disabled.
- `true` - The feature is enabled for all cell types; both combinational and sequential.
- `combinational` - The feature is enabled for combinational cells, but disabled for sequential cells.

Valid values: `false true combinational`

cts_target_max_capacitance

```
cts_target_max_capacitance {auto | double}
```

Default: `auto`

Data_type: string, read/write

Specifies the target maximum capacitive load to allow during clock tree synthesis. This attribute specifies a maximum (combined pin and wire) capacitance that the clock tree synthesis algorithm will allow any given library pin to drive in a given clock tree when driving a given net_type. It is specified in library units. It currently only constrains the primary delay corner capacitance values - other delay corners can be specified but will not be constrained. This attribute is applied in addition to the `max_capacitance` constraints read from the liberty library data - the tightest (lowest) of the constraint specified by this attribute and the constraint present in the liberty data will be used. It also does not apply at the root pins of clock trees - to constrain those nets the `cts_source_max_capacitance` CTS attribute should be used instead.

Valid values: `auto | double`

cts_target_max_capacitance_leaf

```
cts_target_max_capacitance_leaf {auto | double}
```

Default: `auto`

Data_type: string, read/write

The target maximum capacitive load to allow during clock tree synthesis. This attribute specifies a maximum (combined pin and wire) capacitance that the clock tree synthesis algorithm will allow any given library pin to drive in a given clock tree when driving a given net_type. It is specified in library units. It currently only constrains the primary delay corner capacitance values - other delay corners can be specified but will not be constrained. This attribute is applied in addition to the `max_capacitance` constraints read from the liberty library data - the tightest (lowest) of the constraint specified by this attribute and the constraint present in the liberty data will be used. It also does not apply at the root pins of clock trees - to constrain those nets the `cts_source_max_capacitance` CTS attribute should be used instead. This attribute only applies to the leaf net type.

Valid values: `auto | double`

cts_target_max_capacitance_top

```
cts_target_max_capacitance_top {auto | double}
```

Default: `auto`

Data_type: string, read/write

The target maximum capacitive load to allow during clock tree synthesis. This attribute specifies a maximum (combined pin and wire) capacitance that the clock tree synthesis algorithm will allow any given library pin to drive in a given clock tree when driving a given net_type. It is specified in library units. It currently only constrains the primary delay corner capacitance values - other delay corners can be specified but will not be constrained. This attribute is applied in addition to the `max_capacitance` constraints read from the liberty library data - the tightest (lowest) of the constraint specified by this attribute and the constraint present in the liberty data will be used. It also does not apply at the root pins of clock trees - to constrain those nets the `cts_source_max_capacitance` CTS attribute should be used instead. This attribute only applies to the top net type.

Valid values: `auto | double`

cts_target_max_capacitance_trunk

```
cts_target_max_capacitance_trunk {auto | double}
```

Default: `auto`

Data_type: string, read/write

The target maximum capacitive load to allow during clock tree synthesis. This attribute specifies a maximum (combined pin and wire) capacitance that the clock tree synthesis algorithm will allow any given library pin to drive in a given clock tree when driving a given net_type. It is specified in library units. It currently only constrains the primary delay corner capacitance values - other delay corners can be specified but will not be constrained. This attribute is applied in addition to the `max_capacitance` constraints read from the liberty library data - the tightest (lowest) of the constraint specified by this attribute and the constraint present in the liberty data will be used. It also does not apply at the root pins of clock trees - to constrain those nets the `cts_source_max_capacitance` CTS attribute should be used instead. This attribute only applies to the trunk net type.

Valid values: `auto | double`

cts_target_max_stage_delay_sigma

```
cts_target_max_stage_delay_sigma { double | ignore}
```

Default: `ignore`

Data_type: string, read/write

The max per-stage statistical on-chip variation (SOCV) delay sigma target to use for CTS.

cts_target_max_transition_time

```
cts_target_max_transition_time {default | auto | ignore | double}
```

Default: `default`

Data_type: string, read/write

The target slew used for clock tree synthesis. This attribute specifies a maximum slew time that the clock tree synthesis algorithm will allow in this clock tree, in library units. 'default' means 'auto' in primary half corner and 'ignore' in other half corners. If set to 'auto', CTS picks an appropriate value based on the collection of allowed buffer sizes and library parameters, although this may not give optimal quality of results. If set to 'ignore', CTS does not constrain the corner. The tightest (lowest) of the constraint specified by this attribute and the constraint present in the liberty data (`max_transition`) will be used.

Valid values: `default | auto | ignore | double`

cts_target_max_transition_time_leaf

```
cts_target_max_transition_time_leaf {default | auto | ignore | double}
```

Default: `default`

Data_type: string, read/write

The target slew used for clock tree synthesis. This attribute specifies a maximum slew time that the clock tree synthesis algorithm will allow in this clock tree, in library units. 'default' means 'auto' in primary half corner and 'ignore' in other half corners. If set to 'auto', CTS picks an appropriate value based on the collection of allowed buffer sizes and library parameters, although this may not give optimal quality of results. If set to 'ignore', CTS does not constrain the corner. This attribute only applies to the leaf net type.

Valid values: `default | auto | ignore | double`

cts_target_max_transition_time_top

```
cts_target_max_transition_time_top {default | auto | ignore | double}
```

Default: `default`

Data_type: string, read/write

The target slew used for clock tree synthesis. This attribute specifies a maximum slew time that the clock tree synthesis algorithm will allow in this clock tree, in library units. 'default' means 'auto' in primary half corner and 'ignore' in other half corners. If set to 'auto', CTS picks an appropriate value based on the collection of allowed buffer sizes and library parameters, although this may not give optimal quality of results. If set to 'ignore', CTS does not constrain the corner. This attribute only applies to the top net type.

Valid values: `default | auto | ignore | double`

cts_target_max_transition_time_trunk

`cts_target_max_transition_time_trunk {default | auto | ignore | double}`

Default: `default`

Data_type: string, read/write

The target slew used for clock tree synthesis. This attribute specifies a maximum slew time that the clock tree synthesis algorithm will allow in this clock tree, in library units. 'default' means 'auto' in primary half corner and 'ignore' in other half corners. If set to 'auto', CTS picks an appropriate value based on the collection of allowed buffer sizes and library parameters, although this may not give optimal quality of results. If set to 'ignore', CTS does not constrain the corner. This attribute only applies to the trunk net type.

Valid values: `default | auto | ignore | double`

cts_target_skew

`cts_target_skew { default | auto | ignore | double}`

Default: `default`

Data_type: string, read/write

This specifies the target skew for clock tree balancing. This may be set to a numeric value, or one of '`auto`', '`ignore`' or '`default`'.

If set to '`auto`' this indicates that an appropriate skew target should be computed.

If set to '`ignore`' this indicates that skew should not be balanced for this corner/path combination.

If unspecified then the value of this attribute is '`default`'.

If the value of the attribute is 'default' the target skew for late delays in the primary delay corner is interpreted as 'auto' and as '`ignore`' otherwise.

Valid values: `default | auto | ignore | double`

cts_timing_connectivity_based_skew_groups_balance_master_clocks

`cts_timing_connectivity_based_skew_groups_balance_master_clocks {true | false}`

Default: `false`

Data_type: bool, read/write

This attribute has no effect if `cts_timing_connectivity_based_skew_groups` is set to `'off'`.

With timing connectivity based skew groups enabled, this attribute controls how CTS will address the synchrony of disjoint (i.e. non-overlapping) master (i.e. non-generated) SDC clocks.

When this attribute is set to `false`, all disjoint master clocks are assumed to be mutually asynchronous. The skew groups generated by `create_clock_tree_spec` will not constrain the sinks of two such clocks to be balanced together.

When this attribute is set to `true`, the synchrony of disjoint master clocks is determined by consulting the SDC assertions in accordance with the value of attribute `cts_timing_connectivity_based_skew_groups`.
Additional skew groups will be generated by `create_clock_tree_spec`, such that the sinks of each pair of synchronous disjoint master clocks will be balanced together by CTS.

Valid values `true false`

cts_timing_connectivity_based_skew_groups

`cts_timing_connectivity_based_skew_groups` *`string`*

Default: `off`

Data_type: string, read/write

SDC false path assertions may render a generated SDC clock asynchronous to its master clock. Under such a condition, the sinks of the generated clock need not be balanced against the sinks of the master clock: there are no timing paths between the two clocks that can be affected by inter-clock skew. This balancing 'relaxation' is realized by adjusting the skew groups that are generated for the two clocks.

This attribute specifies which SDC assertions are considered when deciding whether a generated clock is asynchronous to its master. Valid values for this attribute are as follows:

- `off`: In this mode, every generated SDC clock is treated as being synchronous to its master SDC clock.

- `clock_false_path`: In this mode, only clock/clock `set_false_path` and `set_clock_group` assertions are considered.

Valid values: `off clock_false_path`

cts_timing_connectivity_info

`cts_timing_connectivity_info` *`string`*

Default: ""

Data_type: string, read/write

This attribute is populated by clock spec creation when timing connectivity based skew groups are enabled. Configuration of this attribute will be written to the generated spec. It documents the clock/clock balancing relationships that were determined for the production of timing connectivity based skew groups. This attribute is documenting only.

If timing connectivity based skew groups are not enabled, this attribute is not populated by spec creation.

The value of the attribute is a set of nested Tcl dictionaries, four levels in total. The top-level dictionary indicates the information category. Currently there is only one such category: 'clock_relationships'. The
second level is keyed on constrained mode name; the lowest level dictionaries are both keyed on SDC clock name. These keys taken together identify two SDC clocks in a given constraint mode. The leaf value specifies the resolved balancing relationships between the two SDC clocks. It is an enum with the following members:

- direct - The two clocks are determined to belong to the same clock group and must be balanced together.
- indirect - The two clocks do not belong to the same clock group; however by transitive closure they must balance together.

Clocks in a direct or indirect balancing relationship will share one or more skew groups so as to ensure that their sinks are balanced together by CTS. Missing entries should be taken as 'need not balance'; i.e. the pair of SDC clocks have neither a direct nor indirect balancing relationship.

The indirect balancing relationship can be explored in more detail using this same Tcl dictionary to probe out the transitive closure over the direct balancing relationships that link two indirect balance clocks together. For example, if clock A directly balances with clock B only, and clock B directly balances with clock C only; then clock A indirectly balances with clock C.

Below is an example dict value, corresponding to clock groups `{clkA clkB} {clkB clkC}` in constraint mode `cm1`, and clock groups `{clkP clkQ} {clkR clkS}` in constraint mode `cm2`.

```
{clock_relationships \
  {cm1 {clkA {clkB direct clkC indirect} \
       clkB {clkA direct clkC direct} \
       clkC {clkA indirect clkB direct}} \
   cm2 {clkP {clkQ direct} \
       clkQ {clkP direct} \
       clkR {clkS direct} \
       clkS {clkR direct}}}}
```

Notice that although they are in separate clock groups, `clkA` and `clkC` nevertheless indirectly balance. This is due to the fact that `clkB` is present in both clock groups. Note also that we omit information about the relationship between `clkP` and `clkS`: they are 'need not balance'.

cts_top_fanout_threshold

`cts_top_fanout_threshold` *`number_of_sinks`*

Default: `unset`

Data_type: string, read/write

Minimum number of transitive fanout in the clock tree for a net to be routed as a top net. Nets with at least this many sinks in their transitive fanout in the clock tree will have the special routing rules applied to them.

Valid values: `integer`

cts_update_clock_latency

`cts_update_clock_latency {true | false}`

Default: `true`

Data_type: bool, read/write

Determine whether to update IO latencies within `ccopt_design`.

Valid values: `true, false`

cts_use_inverters

`cts_use_inverters {auto | true | false}`

Default: `auto`

Data_type: string, read/write

Specifies whether clock tree synthesis should prefer to use inverters rather than buffers when balancing the clock tree. If set to `true`, CTS will use inverters for the clock tree balancing process. If set to `false`, CTS will use the minimum number of levels of inverters required to maintain logical correctness. If set to `auto` (the default) CTS will use what it considers to be the best combination of buffers and inverters to get optimal quality of results.

Valid values: `auto | true | false`

cts_use_receiver_model_capacitance_for_drv

`cts_use_receiver_model_capacitance_for_drv {auto | true | false}`

Default: `false`

Data_type: bool, read/write

When this attribute is set to `auto`, CTS will use receiver model capacitance for drv if the attribute, `timing_report_use_receiver_model_capacitance` is `true`. When set to `true`, CTS will use receiver model capacitance for drv even if `timing_report_use_receiver_model_capacitance` is `false`. By default, this attribute is set to `false`, which means that CTS will not use receiver model capacitance for drv.

Valid values: `auto true false`

flexible_htrees

`flexible_htrees` *`list of flexible_htree`*

Data_type: flexible_htree*, read-only

Returns the list of flexible h-trees in the design.

Valid values: `list of flexible_htree`

skew_groups

`skew_groups` *`list of skew_groups`*

Data_type: skew_group*, read-only

Returns the list of skew groups in the design.

Valid values: `list of skew_group`

delete_clock_tree_drivers

```
delete_clock_tree_drivers
[-help]
[-force]
```

This command removes buffers and inverter pairs from the clock trees. If used without specifying any parameters, it deletes all repeaters except those that have fixed or dont_touch attributes that are assigned either by the user or by `ccopt_design`. To delete all repeaters, specify the `-force` parameter.

Note: After running the command, there may be remaining inverters in the clock tree that are required to preserve an inversion. The remaining inverters may not be the same as the ones that were originally introduced for inversion, which means that the naming can differ.

Parameter

`-help`	Prints a brief description that includes type and default information for each `delete_clock_tree_drivers` parameter. For a detailed description of the command and its parameter, use the man command: `man delete_clock_tree_drivers`.
`-force`	Deletes all repeaters even if they have fixed or dont_touch attributes. *Data_type*: bool, optional

Related Command

- `ccopt_design`

delete_clock_tree_source_group

```
delete_clock_tree_source_group
[-help]
pattern
[-regexp]
```

This command deletes all clock tree source groups matching the specified pattern.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `delete_clock_tree_source_group` parameter. For a detailed description of the command and all of its parameters, use the man command: `man delete_clock_tree_source_group`.
`pattern`	Specifies the patterns for clock tree source groups to delete. This parameter is required. *Data_type*: string, required
`-regexp`	Specifies whether or not to use regular expression matching. *Data_type*: bool, optional

Example

- Use the following command to delete all clock tree source groups starting with `"clk"`:
  ```
  delete_clock_tree_source_group clk*
  ```

Related Information

- `create_clock_tree_spec`
- `create_clock_tree`
- `create_clock_tree_source_group`
- Clock Tree Synthesis chapter in the *Innovus Stylus Common UI User Guide*

delete_clock_tree_spec

```
delete_clock_tree_spec
[-help]
[-preserve_sink_insertion_delays]
```

This command deletes all the skew group definitions and other clock tree synthesis configuration information. However, you can preserve the pin insertion delays on clock tree sinks by specifying the `-preserve_sink_insertion_delays` parameter.

Parameter

`-help`	Outputs a brief description of the `delete_clock_tree_spec` parameter.
`-preserve_sink_insertion_delays`	When specified, preserves pin insertion delays currently configured on clock tree sinks. *Data_type*: bool, optional

Example

- Use the following command to get a list of clock trees:
  ```
  get_db clock_trees
  m_clk m_digit_clk m_dsram_clk m_ram_clk m_rcc_clk m_spi_clk refclk<1> refclk
  ```
- Use the following command to get a list of skew groups:
  ```
  get_db skew_groups
  m_clk/PM_HL_FUNC m_digit_clk/PM_HL_FUNC m_dsram_clk/PM_HL_FUNC m_ram_clk/PM_HL_FUNC m_rcc_clk/PM_HL_FUNC m_spi_clk/PM_HL_FUNC
  refclk/PM_HL_FUNC
  ```
- Use the following command to delete all CTS configuration:
  ```
  delete_clock_tree_spec
  ```
- When you check for clock trees and skew groups after deleting the CTS configuration, the software returns nothing.
  ```
  get_db clock_trees
  nothing returned
  get_db skew_groups
  nothing returned
  ```

Related Information

- `ccopt_design`
- `create_clock_tree_spec`
- `create_clock_tree`
- `get_db`
- The following sections in Clock Tree Synthesis chapter of *Innovus Stylus Common UI User Guide*
  - The Clock Tree Synthesis Engines
  - Flow and Quick Start
  - Concepts and Clock Tree Specification

delete_clock_trees

```
  delete_clock_trees
[-help]
pattern
[-regexp]
```

This command deletes all clock trees matching the specified pattern. It makes no changes to the design.

Note: It is recommended to use the `delete_clock_tree_spec` command to reset the entire CTS configuration. It removes clock tree definition from the in-memory clock tree specification.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `delete_clock_trees` parameter. For a detailed description of the command and all of its parameters, use the man command: man `delete_clock_trees`.
`pattern`	Specifies the patterns for clock trees to delete. This parameter is required. *Data_type*: string, required
`-regexp`	Specifies whether or not to use regular expression matching. *Data_type*: bool, optional

Example

- The following command deletes all clock trees whose names start with "ref":
  ```
  delete_clock_trees ref*
  ```
  The software gives the following message:
  ```
  Un-defining clock tree refclk.
  ```

Related Information

- `ccopt_design`
- `create_clock_tree`
- `create_clock_tree_spec`
- `delete_clock_tree_spec`
- The following section in Clock Tree Synthesis chapter of *Innovus Stylus Common UI User Guide*
  - Clock Tree Concepts and Specifications

delete_flexible_htrees

```
delete_flexible_htrees
[-help]
pattern
[-regexp]
```

This command deletes all flexible H-trees matching the specified pattern.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `delete_flexible_htrees` parameter. For a detailed description of the command and all of its parameters, use the man command: `man delete_flexible_htrees`.
`pattern`	Specifies the patterns for flexible H-trees to delete. This parameter is required. *Data_type*: string, required
`-regexp`	Specifies whether or not to use regular expression matching. *Data_type*: bool, optional

Example

The following command deletes all flexible trees whose names start with "`htree`":

```
delete_flexible_htrees htree*
```

Related Information

- `ccopt_design`
- `create_flexible_htree`
- `synthesize_flexible_htrees`
- The following section in Clock Tree Synthesis chapter of *Innovus Stylus Common UI User Guide*
  - Flexible H-Tree and Multi-Tap Clock Flow
- For details of the use model for this feature, see the Flexible H-tree and Multi-Tap Clock Flow in Innovus Application Note on the Cadence Online Support website.

delete_skew_groups

```
delete_skew_groups
[-help]
pattern
[-regexp]
```

This command deletes all skew groups matching the specified pattern. It makes no changes to the design.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `delete_skew_groups` parameter. For a detailed description of the command and all of its parameters, use the man command: `man delete_skew_groups`.
`pattern`	Specifies the patterns for skew groups to delete. This parameter is required. *Data_type*: string, required
`-regexp`	Specifies whether or not to use regular expression matching. *Data_type*: bool, optional

Examples

- The following command deletes all skew groups associated with the clock, `m_digit_clk`:
  ```
  delete_skew_groups m_digit_clk*
  ```
- The following command deletes all skew groups whose name contains the word "`ram`":
  ```
  delete_skew_groups ram -regexp
  ```

Related Information

- `create_skew_group`
- `get_db`
- `get_skew_group_path`
- `get_skew_group_delay`
- `report_skew_groups`
- `set_db`
- `update_skew_group`
- The following section in Clock Tree Synthesis chapter of *Innovus Stylus Common UI User Guide*
  - Clock Trees and Skew Groups

get_ccopt_property_common_ui_map

```
get_ccopt_property_common_ui_map
[-help]
property name
```

This command helps you map all CTS properties, public and private, in Legacy Innovus to the corresponding CTS and CCOpt category attributes in Stylus. The use of wildcards is not supported while specifying the name of the property. For example:

```
get_ccopt_property_common_ui_map ccopt_*
[get_ccopt_property_common_ui_map]: The use of a wildcard is not allowed.
```

If the specified legacy CTS property name does not exist, the software gives the following message:

```
get_ccopt_property_common_ui_map add_multiple
No legacy CCOpt property exists matching 'add_multiple'.
```

If the specified legacy CTS property has multiple associated attributes, the software provides the basic, one-line help information for all these attributes:

```
get_ccopt_property_common_ui_map target_max_trans
Attributes:
cts_target_max_transition_time(clock_tree)
# string, read/write, default=top default trunk default leaf default, indices=delay_corner power_domain
cts_target_max_transition_time_leaf(clock_tree)
# string, read/write, default=default, indices=delay_corner power_domain
cts_target_max_transition_time_top(clock_tree)
# string, read/write, default=default, indices=delay_corner power_domain
cts_target_max_transition_time_trunk(clock_tree)
# string, read/write, default=default, indices=delay_corner power_domain
cts_target_max_transition_time(root)
# string, read/write, default=top default trunk default leaf default, indices=delay_corner power_domain
cts_target_max_transition_time_leaf(root)
# string, read/write, default=default, indices=delay_corner power_domain
cts_target_max_transition_time_top(root)
# string, read/write, default=default, indices=delay_corner power_domain
cts_target_max_transition_time_trunk(root)
# string, read/write, default=default, indices=delay_corner power_domain
```

Parameters

`-help`	Outputs a brief description that includes type and default information for each `get_ccopt_property_common_ui_map` parameter. For a detailed description of the command and all of its parameters, use the man command: `man get_ccopt_property_common_ui_map` .
`property name`	Specifies the name of the legacy Innovus CTS property for which the corresponding attribute name is to be retrieved. *Data_type*: string, required

Examples

- The following command retrieves the name of the CTS attribute corresponding to the `add_exclusion_drivers` property:

```
get_ccopt_property_common_ui_map add_exclusion_drivers
Attribute
cts_add_exclusion_drivers(root)
# bool, read/write, default=true, indices={}
```

- The following command retrieves the name of the CCOpt attribute corresponding to the `auto_limit_insertion_delay_factor` property:

```
get_ccopt_property_common_ui_map auto_limit_insertion_delay_factor
Attribute
ccopt_auto_limit_insertion_delay_factor(root)
# double, read/write, default=1.5, indices={}
```

Related Information

- The following chapters in *Innovus Stylus Common UI Text Command Reference:*
  - cts Category Attributes
  - ccopt Category Attributes
- The Clock Tree Synthesis chapter of *Innovus Stylus Common UI User Guide.*

Related Commands

- `get_common_ui_map`

get_skew_group_path

```
get_skew_group_path
[-help]
[-check_type {setup | hold}]
[-delay_corner delay_corner]
[-delay_type {early | late}]
[-show_generator_paths]
-skew_group skew_group_name
[-longest  | -shortest]
[-summarize_stage_depth cell_type_list]
[-virtual_delays_only  | -real_delays_only]
[-wire_delays_only  | -cell_delays_only]
[-sink pin | -below pin]
```

This command returns the path delay information for the specified skew group. By default, this command returns the longest source-to-sink path in the specified skew group, which is the path with the largest delay. Paths are returned as a list of pins. You can use `-shortest` to return the shortest source-to-sink path rather than the longest.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `get_skew_group_path` parameter. For a detailed description of the command and all of its parameters, use the man command: man `get_skew_group_path`.
`-below pin`	Specifies that CTS will only consider paths below this pin. *Data_type*: (pin|port), required
`-cell_delays_only`	Specifies that CTS will only consider cell delays (ignoring wire delays). By default both cell and wire delays are used to calculate delays. *Data_type*: bool, optional
`-check_type {setup | hold}`	
	Specifies whether the check type for the specified skew group should be setup or hold. If a check type is specified, but the `-delay_type` parameter is not specified, the software behaves in the following manner: • `-check_type -setup` will mean `-delay_type -late` • `-check_type -hold` will mean `-delay_type -early` **Note**: If a check type is not specified, the software will always look at what the views permit, preferring the “setup” view if there is a choice. *Data_type*: enum, optional
`-delay_corner delay_corner`	
	Specifies the delay corner to use when calculating delays. If this parameter is not specified, the software will use the delay corner specified using the CTS attribute, `cts_primary_delay_corner`. However, if no valid delay corner is found, the software will error out. **Note**: For detailed help on the attribute, `cts_primary_delay_corner`, see the cts Category Attributes chapter. *Data_type*: (delay_corner), optional
`-delay_type {early | late}`	
	Specifies whether to use early or late half delays. When `early` is specified, CTS uses only early path timing data to calculate path delay, for the clock edge specified. When `late` is specified, CTS uses only late path timing data to calculate path delay, for the clock edge specified. **Note**: If the delay type is not specified but the check type is specified as “setup”, the software will consider the delay type as “late”. If the check type is “hold”, the software will consider the delay type as “early”. *Data_type*: enum, optional

`-longest`	Specifies that CTS will return the longest path delay within the skew group. This is the default option. *Data_type*: bool, optional
`-real_delays_only`	Specifies that CTS will only consider the real path delays in the design. It will not include the virtual path delays added by trial CTS to resolve skew constraints. *Data_type*: bool, optional
`-shortest`	Specifies that CTS will return the shortest source-to-sink path within the skew group (that is, the path with the smallest source-to-sink delay). By default, this command returns the longest path. *Data_type*: bool, optional
`-show_generator_paths`	Specifies that CTS follows generated clock tree paths when determining the longest source-to-sink path. By default generated clock tree paths are not followed. *Data_type*: bool, optional
`-sink` *pin*	Specifies that CTS will return the longest path to this pin for the specified skew group. *Data_type*: (pin|port), required)
`-skew_group` *skew_group_name*	
	Specifies the name of the skew group for which path information will be returned. This is a required parameter. *Data_type*: string, required
`-summarize_stage_depth` *cell_type_list*	

	Specifies that the command will consider the number of cells of the specified type(s) in the tree, instead of considering delay. Possible values are buffer, inverter,logic, clock_gate, and generator. You can also specify * to include all cell types.
	For example, the following command shows you the path with most buffers and inverters in the skew group, `PM_HL_FUNC`.
	`get_skew_group_path -summarize_stage_depth {buffer inverter} -longest -skew_group m_dsram_clk/PM_HL_FUNC`
	Note: The * setting is not used in combination with any other cell type. For example, the below is not a valid setting.
	`-summarize_stage_depth {* buffer}`
	Data_type: list, optional

`-virtual_delays_only`	Specifies that CTS will return the shortest path delay or longest path delay only considering the virtual path delays (added by trial balancing) in the design. It will not include real path delays added by clustering buffers and existing cells in the design. *Data_type*: bool, optional
`-wire_delays_only`	Specifies that CTS will only consider wire delays (ignoring cell delays). By default both cell and wire delays are used to calculate delays. *Data_type*: bool, optional

Examples

- Use the following command to retrieve the list of skew groups:
  ```
  get_db skew_groups
  m_clk/PM_HL_FUNC m_digit_clk/PM_HL_FUNC m_dsram_clk/PM_HL_FUNC m_ram_clk/PM_HL_FUNC m_rcc_clk/PM_HL_FUNC m_spi_clk/PM_HL_FUNC
  refclk/PM_HL_FUNC
  ```
- Use the following command to retrieve the path delay information for the specified skew group:
  ```
  get_skew_group_path -skew_group m_dsram_clk/PM_HL_FUNC
  TEST_CONTROL_INST/g139/ZN 0.000 RAM_256x16_TEST_INST/CPF_LS_160_m_dsram_clk/I 0.000 RAM_256x16_TEST_INST/CPF_LS_160_m_dsram_clk/Z
  16817.000 RAM_256x16_TEST_INST/RAM_256x16_INST/CLK 17463.000
  ```
- Use the following command to retrieve the paths with most buffers and inverters in the skew group, `PM_HL_FUNC`:
  ```
  get_skew_group_path -summarize_stage_depth {buffer inverter} -longest -skew_group m_dsram_clk/PM_HL_FUNC
  TEST_CONTROL_INST/AZ_cid_BUF_m_dsram_clk_G0_L1_1/Z RAM_256x16_TEST_INST/CPF_LS_160_m_dsram_clk/I
  RAM_256x16_TEST_INST/CPF_LS_160_m_dsram_clk/Z CORE_INST/CORE_MACH_INST/RC_CG_HIER_INST29/AZ_csf_BUF_m_dsram_clk_G0_L3_1/I
  CORE_INST/CORE_MACH_INST/RC_CG_HIER_INST29/AZ_csf_BUF_m_dsram_clk_G0_L3_1/Z
  CORE_INST/CORE_MACH_INST/RC_CG_HIER_INST29/AZ_csf_BUF_m_dsram_clk_G0_L4_1/I
  ```

```
CORE_INST/CORE_MACH_INST/RC_CG_HIER_INST29/AZ_csf_BUF_m_dsram_clk_G0_L4_1/Z
CORE_INST/CORE_MACH_INST/RC_CG_HIER_INST29/AZ_csf_BUF_m_dsram_clk_G0_L5_1/I
CORE_INST/CORE_MACH_INST/RC_CG_HIER_INST29/AZ_csf_BUF_m_dsram_clk_G0_L5_1/Z
CORE_INST/CORE_MACH_INST/RC_CG_HIER_INST29/AZ_csf_BUF_m_dsram_clk_G0_L6_1/I
CORE_INST/CORE_MACH_INST/RC_CG_HIER_INST29/AZ_csf_BUF_m_dsram_clk_G0_L6_1/Z RAM_256x16_TEST_INST/RAM_256x16_INST/CLK
```

Related Information

- create_skew_group
- delete_skew_groups
- get_db
- get_skew_group_delay
- report_skew_groups
- set_db
- update_skew_group
- The following section in Clock Tree Synthesis chapter of *Innovus Stylus Common UI User Guide*
  - Manual Setup and Adjustment of the Clock Specification

get_clock_tree_capacitance

```
get_clock_tree_capacitance
[-help]
-pin pinName
[-delay_corner corner]
[-delay_type {early | late}]
[-edge {rise | fall | both}]
[-load | -wire]
```

This command retrieves the capacitance on the specified pin. When you run the command without specifying the type of capacitance to be retrieved, load or wire, then the command retrieves the total capacitance on a pin, which includes both the wire capacitance of nets connected to the pin, and the load capacitance of pins connected to those nets. Because the capacitance varies depending on the timing data used to calculate delays, this command provides options such as rise, fall, early, and late that let you specify the exact timing criteria to use when calculating capacitances.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `get_clock_tree_capacitance` parameter. For a detailed description of the command and all of its parameters, use the man command: `man get_clock_tree_capacitance`.
`-delay_corner corner`	Specifies the delay corner of the specified pin for which capacitance information is provided. By default, if delay corner is not specified, CCOpt uses the CTS primary corner as configured by the `cts_primary_delay_corner <corner>` attribute. If this is not set, it will use the default setup analysis view as configured by the `set_default_view -setup <view>` command. *Data_type*: (delay_corner), optional
`-delay_type {early | late}]`	Specifies whether to use early or late half delays. When `early` is specified, CCOpt uses only early path timing data to calculate capacitances, for the clock edge specified. When `late` is specified, CCOPT uses only late path timing data to calculate capacitances, for the clock edge specified. By default, if neither early or late are specified, CCOpt will use late half delays to calculate capacitances. *Data_type*: enum, optional
`-edge {rise | fall | both}`	
	Specifies the edge of the specified pin for which capacitance is to be calculated. When `fall` is specified, CCOpt uses only `falling` clock edge timing data to calculate capacitances. When `rise` is specified, CCOpt uses only `rising` clock edge timing data to calculate capacitances. By default, if neither `rise` or `fall` are specified or if both is specified, CCOpt uses both rising and falling clock edge timing data to calculate capacitances. *Data_type*: enum, optional
`-load | -wire`	When `-load` is specified, the command returns the driven load capacitance value of the specified pin. When `-wire` is specified, the command returns the driven wire capacitance value of the specified pin. The two options are mutually exclusive. *Data_type*: bool, optional
`-pin pinName`	Specifies the pin for which the load capacitance value is to be returned.This is a required parameter. *Data_type*: (pin|port), required

Example

- Use the following command to return the pins for the specified object:

```
get_property [get_pins -of_object RESULTS_CONV_INST/upper770/g54] hierarchical_name
get_pins -of_object RESULTS_CONV_INST/upper770/g54
RESULTS_CONV_INST/upper770/g54/ZN RESULTS_CONV_INST/upper770/g54/I
```

- Use the following command to return the capacitance information for the specified pin:

```
get_clock_tree_capacitance RESULTS_CONV_INST/upper770/g54/ZN
```

  The software returns the following:

```
0.0160777831078
```

- Use the following command to return the wire capacitance information for the specified pin:

```
get_clock_tree_capacitance TEST_CONTROL_INST/g145/ZN -wire
```

  The software returns the following:

```
0.00874637126923
```

get_clock_tree_effective_max_capacitance

```
get_clock_tree_effective_max_capacitance
[-help]
-pin
[-delay_corner corner]
[-delay_type {early | late}]
[-source]
[-value]
```

Returns the value of the frequency-dependent effective maximum capacitance constraint that the software will apply at a given pin in the clock tree. By default, the software returns the following information:

- The actual value of the maximum effective capacitance or constrained maximum capacitance at the specified pin
- The source of the constraint

Parameters

`-help`	Outputs a brief description that includes type and default information for each `get_clock_tree_effective_max_capacitance` parameter. For a detailed description of the command and all of its parameters, use the man command: `man get_clock_tree_effective_max_capacitance`.
`-delay_corner` *corner*	Specifies the delay corner of the specified pin for which effective maximum capacitance information is provided. By default, if delay corner is not specified, CCOpt uses the CTS primary corner as configured by the `cts_primary_delay_corner` *delay_corner_name* attribute. If this is not set, it will use the default setup analysis view as configured by the command, `set_default_view -setup` *newDefaultSetupView*. *Data_type*: (delay_corner), optional
`-delay_type {early | late}`	Specifies whether to use early or late half delays. When `early` is specified, CCOpt uses only early path timing data to calculate effective maximum capacitance, for the specified clock edge. When `late` is specified, CCOpt uses only late path timing data to calculate effective maximum capacitance, for the specified clock edge. By default, if neither `early` or `late` are specified, CCOpt will use late half delays to calculate effective maximum capacitance. *Data_type*: enum, optional
`-pin`	Specifies the pin for which the effective maximum capacitance value is to be returned. This is a required parameter. *Data_type*: (pin|port), required
`-source`	Specifies the source of the effective maximum capacitance. It is a boolean type. The reported source may be any of the following: • `library_or_sdc`: Specifies that the most constraining capacitance came from a non-frequency dependent library constraint on a library cell pin or SDC override. This can also apply to clock roots based on the `cts_source_driver` attribute where a clock root can be modeled as a specific type of library cell. • `freq_dep_library`: Specifies that the most constraining capacitance value came from a frequency-dependent library constraint. As per `library_or_sdc`, this can also apply to clock roots via library cell modeling. • `target_max_capacitance`: Specifies that the most constraining capacitance value came from an explicit `cts_target_max_capacitance` attribute value. • `source_max_capacitance`: Specifies that the most constraining capacitance value came from the `cts_source_max_capacitance`attribute. • `computed_source_max_capacitance`: Specifies that the most constraining capacitance value was computed by CCOpt for a clock root based on a library cell and a transition target. *Data_type*: bool, optional
`-value`	Reports the effective maximum capacitance value of the specified pin. *Data_type*: bool, optional

Examples

- Use the following command to return the pins for the specified object, `TEST_CONTROL_INST/g139`:

  ```
  get_pins -of_object TEST_CONTROL_INST/g139
  TEST_CONTROL_INST/g139/A1 TEST_CONTROL_INST/g139/A2 TEST_CONTROL_INST/g139/B TEST_CONTROL_INST/g139/ZN
  ```

- Use the following command to return the source of the effective maximum capacitance value:

  ```
  get_clock_tree_effective_max_capacitance TEST_CONTROL_INST/g139/ZN -source
  ```

  The software returns the following information:

  ```
  library_or_sdc_constraint
  ```

- Use the following command to return the effective maximum capacitance value:

  ```
  get_clock_tree_effective_max_capacitance TEST_CONTROL_INST/g139/ZN -value
  ```

  The software returns the following:

  ```
  0.0853199996948
  ```

get_clock_tree_transition_time

```
get_clock_tree_transition_time
[-help]
[-delay_corner corner]
[-delay_type {early | late}]
[-edge {rise | fall | both}]
-pin pinName
```

This command returns the slew information for the user-specified pin.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `get_clock_tree_transition_time` parameter. For a detailed description of the command and all of its parameters, use the man command: `man get_clock_tree_transition_time`.
`-delay_corner corner`	
	Specifies the delay corner of the specified pin to be queried for slew information. By default, if delay corner is not specified, CCOpt uses the CTS primary corner as configured by the `cts_primary_delay_corner corner` attribute. If this is not set, it will use the default setup analysis view as configured by the `set_default_view -setup view` command. *Data_type*: (delay_corner), optional
`-delay_type {early | late}]`	
	Specifies whether to use early or late half delays. When `early` is specified, CCOpt uses only early path timing data to calculate slew, for the clock edge specified. When `late` is specified, CCOpt uses only late path timing data to calculate slew, for the clock edge specified. By default, if neither early or late are specified, CCOpt will use late half delays to calculate slew information. *Data_type*: enum, optional
`-edge { rise | fall | both }`	
	Specifies the edge of the specified pin for which slew information is to be calculated. When `fall` is specified, CCOpt uses only `falling` clock edge timing data to calculate slew. When `rise` is specified, CCOpt uses only `rising` clock edge timing data to calculate slew. By default, if neither rise or fall are specified, CCOpt uses both rising and falling clock edge timing data to calculate slew. *Data_type*: enum, optional
`-pin pinName`	Specifies the pin for which the slew information is provided. This is a required parameter. *Data_type*: (pin|port), required

Examples

- Use the following command to get a list of clock trees:
  ```
  get_clock_tree_transition_time *
  PLLCLK_INST TEST_CONTROL_INST/CPF_LS_101_clk TEST_CONTROL_INST/g145 TEST_CONTROL_INST/g144 TEST_CONTROL_INST/g136...
  ```

- Use the following command to get the list of pins in the specified clock tree:
  ```
  get_pins TEST_CONTROL_INST/g145/*
  TEST_CONTROL_INST/g145/ZN TEST_CONTROL_INST/g145/A2 TEST_CONTROL_INST/g145/A1
  0x5d5
  ```

- Use the following command to get the slew value for the specified pin in the specified clock tree:
  ```
  get_clock_tree_transition_time TEST_CONTROL_INST/g145/ZN
  ```

  The software returns the following value for the slew for the pin ZN:
  ```
  0.2228
  ```

get_cts_graph_traversal

```
get_cts_graph_traversal
[-help]
[-fanin]
[-fanout]
[-of pin | port | inst*]
[-skew_group skew_group]
[-skip buffers inverters drivers clock_gates logics generators generated_roots all]
[-transitive]
```

This command traverses the clock tree graph, starting at the specified pins, ports, or insances and traversing to their fanin or fanout, returning a list of the resulting pins or port objects. Instance types can be skipped over, and can be restricted to a specified skew group.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `get_cts_graph_traversal` parameter. For a detailed description of the command and all of its parameters, use the man command: `man get_cts_graph_traversal`.
`-fanin`	Traverse graph towards source pins or ports of the specified objects. *Data_type*: bool, optional
`-fanout`	Traverse graph towards load pins or ports of the specified objects. *Data_type*: bool, optional
-of pin | port | inst*	
	Specifies a list of pins, ports, or instances (objects, names, or wildcard-names) to start traversing from. The direction of traversal is controlled by `-fanin` or `-fanout`. *Data_type*: (pin|port|inst)*, optional
`-skew_group` *skew_group_name*	Skew group object or name. If specified, restricts traversal to instances in the skew group. *Data_type*: string, optional
`-skip` *buffers inverters drivers clock_gates logics generators generated_roots all*	
	Specifies the types of nodes that should be skipped past to their fanin/fanout during the traversal. You can specify one or more of the following: • buffers • inverters • drivers (equivalent to buffers and inverters) • clock_gates • logics • generators • generated_roots • all (equivalent to listing all the above, and also equivalent to `-transitive`) *Data_type*: list, optional
`-transitive`	Skip all types and get sinks (if `-fanout`) or clock roots (if `-fanin`). This is equivalent to the `-skip all` option. *Data_type*: bool, optional

Examples

- ```
  set simple_fanout [get_cts_graph_traversal -fanout -of CTS_ccd_BUF_clock_G0_L1_1]
  set simple_fanout [get_cts_graph_traversal -fanout -of [get_db insts CTS_ccd_BUF_clock_G0_L1_1]]
  set fanin_with_skip [get_cts_graph_traversal -fanin -skip {buffers generators logics} -of [get_db pins CTS_*/A]]
  foreach pin_or_port_name [get_db $fanin_with_skip .name] {
  puts "Got $pin_or_port_name"
  }
  ```
  ```

get_lib_clock_tree_path_delay

```
get_lib_clock_tree_path_delay
[-help]
-base_pin base_pin_name
[-delay_type {max | min}]
[-edge {rise | fall}]
[-mode analysis_mode_name]
[-power_domain power_domain]
-transition transition_time
-view view_name
```

This command retrieves information about the Liberty `max_clock_tree_path` and `min_clock_tree_path` delay attributes for a library pin. You can then use Tcl to apply the command options, as required. The value returned by the `get_lib_clock_tree_path_delay` command is in the default system units, such that the user can pass to the `set_clock_latency` command directly. The same units are used for the `-transition` value.

If the library pin does not have a corresponding `min/max_clock_tree_path` delay attribute, the command returns a value of `0`. If other errors are encountered, the command fails.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `get_lib_clock_tree_path_delay` parameter. For a detailed description of the command and all of its parameters, use the man command: `man get_lib_clock_tree_path_delay`.
`-base_pin base_pin_name`	
	Specifies the name of the base pin for which the `min/max_clock_tree_path` attribute value is to be retrieved. This is a required parameter. *Data_type*: (base_pin), required
`-delay_type {max | min}`	
	Specifies whether to use `max` or `min` delays. You can specify either `max` or `min` to retrieve the `max_clock_tree_path` and `min_clock_tree_path` delay values, respectively. *Default*: `max` *Data_type*: enum, optional
`-edge {rise | fall}`	
	Specifies whether the delay should be retrieved for the rising edge or the falling edge of the signal. *Default*: `rise` *Data_type*: enum, optional
`-mode analysis_mode_name`	Specifies the Liberty timing library mode value to be used while retrieving the delay attributes for the library pin. If no mode definition is available, the default look-up table (LUT) is used. The detailed use model is provided below. **User-Specified Item**: **Software Behavior** • mode + delay_type: Use the specified mode and the specified delay_type LUT • mode: Use the specified mode and the first max LUT • delay_type: Use the first specified delay type LUT • No mode and no delay type: Use the first max LUT For more information about setting modes, see `set_mode`. *Data_type*: string, optional
`-power_domain power_domain`	
	Specifies the name of the power domain. This parameter is only required to be specified when the same base cell has different models between different power domains. *Data_type*: (power_domain), optional

`-transition transition_time`	
	Specifies the input transition time. This is a required parameter. *Data_type*: float, required, min=0, max=99
`-view view_name`	
	Specifies the analysis view name. This is a required parameter. *Data_type*: (analysis_view), required

Examples:

- Use the following command to retrieve the value of `max_clock_tree_path` delay at the rising edge of the specified pin:
  ```
  get_lib_clock_tree_path_delay -base_pin macro_lib_cell/CLK -view setup_func_m40 -transition 0.1
  ```
- Use the following command to retrieve the value of `min_clock_tree_path` delay at the falling edge of the specified pin:
  ```
  get_lib_clock_tree_path_delay -base_pin macro_lib_cell/CLK -view setup_func_m40 -delay_type min -edge fall -transition 0.1
  ```
- Use the following command to retrieve the value of `max_clock_tree_path` delay when `macro_lib_cell` is specified only in the `setup_func_m40c` view of `PD_TOP` power domain:
  ```
  get_lib_clock_tree_path_delay -base_pin macro_lib_cell/CLK -view setup_func_m40 -power_domain PD_TOP -transition 0.1
  ```

Related Information

- `convert_lib_clock_tree_latencies`
- `set_clock_latency`
- The following section in Clock Tree Synthesis chapter of *Innovus Stylus Common UI User Guide*
  - Converting Library Path Delays to Clock Latencies

get_skew_group_delay

```
get_skew_group_delay
[-help]
[-check_type {setup | hold}]
[-delay_corner delay_corner]
[-delay_type {early | late}]
[-edge {rise | fall}]
[-exclude_pin_insertion_delays]
-skew_group skew_group_name
[-skew | -longest | -shortest ]
[-path_length distance_metric | -summarize_stage_depth cell_type_list]
[-all_active_sinks | -to pin | -from pin | -through pin ]
[-virtual_delays_only | -real_delays_only ]
[-wire_delays_only | -cell_delays_only ]
[-delay_corner delay_corner ]
```

By default, this command returns the longest source-to-sink delay for the specified skew group. Alternatively, you can use `-shortest` to return the shortest source-to-sink delay, or `-skew` to return the difference between the shortest and longest source-to-sink delay.

Optional arguments let you restrict the set of pins considered. For example, use `-to` to return the longest delay to the specified sink pin, use `-delay_type` and `-delay_corner` to specify how delays are calculated, and use `-wire_delays_only` to specify which delays are considered.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `get_skew_group_delay` parameter. For all of its parameters, use the man command: `man get_skew_group_delay`.
`-all_active_sinks`	Specifies that the result should be a tcl dict of pairs of sink and delay, describing delays for all the active sinks of the `latency_a sink_b latency_b ... sink_n latency_n}`.
`-cell_delays_only`	Specifies that CTS only considers cell delays when calculating delays and ignores wire delays. *Data_type*: bool, optional
`-check_type {setup | hold}`	
	Specifies whether to use setup or hold timing as the check type to use for the specified skew group. If a check type is specified, but the `-delay_type` parameter is not specified, the software behaves in the following ma • `-check_type -setup` will mean `-delay_type -late` • `-check_type -hold` will mean `-delay_type -early` **Note**: If a check type is not specified, the software will always look at what the views permit, preferring the “setup” vie *Data_type*: enum, optional
`-delay_corner delay_corner`	
	Specifies the delay corner to use when calculating delays. If this parameter is not specified, the software will use the delay corner specified using the CTS attribute, `cts_primar` corner is found, the software will error out. **Note**: For information about the attribute, `cts_primary_delay_corner`, see the cts Category Attributes chapter. *Data_type*: (delay_corner), optional
`-delay_type {early | late}`	
	Specifies whether to use early or late half delays. When `early` is specified, CTS uses only early path timing data to calculate delay, for the clock edge specified. When timing data to calculate delay, for the clock edge specified. **Note**: If the delay type is not specified but the check type is specified as “setup”, the software will consider the delay t software will consider the delay type as “early”. *Data_type*: enum, optional

`-edge {rise | fall}`	Specifies the edge of the specified pin for which delay information is to be calculated. When `fall` is specified, CTS uses only `falling` clock edge timing data to calculate delay. When `rise` is specified, C data to calculate delay. *Data_type*: enum, optional
`-exclude_pin_insertion_delays`	Specifies that the delay should exclude pin insertion delays.
`-from` *pin*	Specifies the pin to query delays from. This means that only paths from the given pin to a sink of the skew group will (pin|port), optional
`-longest`	Specifies that CTS will return the delay of the longest source-to-sink path within the skew group. This is the default o *Data_type*: bool, optional

`-path_length {half_perim pin_to_pin_steiner routed}`	Specifies that CTS will report by the distance from the root measured using the specified distance metric. You can ch metrics:
	• `half_perim`: The sum of the Manhattan distances between terminals on the path. This is not blockage-aware and
	• `pin_to_pin_steiner`: The sum of lengths calculated using a pin-to-pin steiner route. This is blockage-aware and i
	• `routed`: The sum of pin-to-pin paths calculated on the full steiner for each net. This is blockage-aware but is not su
	The routed metric is best post-CTS as it measures the actual clock path. The other metrics are good for pre-CTS alth CTS. For pre-CTS, the `half_perim` metric is fast to calculate while the other metrics are slow.
	Example:
	`get_skew_group_delay -skew_group skew_group:clk/default_constraint_mode -longest -path_length half_per`
	`get_skew_group_delay -skew_group skew_group:clk/default_constraint_mode -longest -path_length half_per 930.0045`
	Note:
	• You can only specify one metric at a time.
	• You have to specify a skew group and that the length reported will be from the skew group sources to the skew gr clock tree. The calculated length will take account of any cells on the clock path, which may increase the path len The -longest, -shortest, and -skew options work the same. This means that the software looks at all skew group si longest/shortest distance. This sink will not always be the same with the three metrics.
	• This parameter is mutually exclusive with the `-summarize_stage_depth` parameter.
	Data_type: enum, optional

`-real_delays_only`	Specifies that CTS only considers real delays in the design. It will not include the virtual delays added by trial CTS to *Data_type*: bool, optional
`-shortest`	Specifies that CTS will return the delay of the shortest source-to-sink path within the skew group. *Data_type*: bool, optional
`-skew`	Specifies that CTS will return the difference between the shortest and longest source-to-sink delays within the skew *Data_type*: bool, optional
`-skew_group` *skew_group_name*	
	Specifies the name of the skew group for which the delay information will be returned. This is a required parameter. *Data_type*: string, required
`-summarize_stage_depth` *cell_type_list*	

	Specifies that the command will consider the number of cells of the specified type(s) in the tree, instead of considerin inverter, logic, clock_gate, and generator. You can also specify `*` to include all cell types.
	For example, the following command tells you the number of buffers and inverters along the specified path type in the
	`get_skew_group_delay -summarize_stage_depth {buffer inverter} -skew_group m_dsram_clk/PM_HL_FUNC`
	Note: The `*` setting is not used in combination with any other cell type. For example, the below is not a valid setting.
	`-summarize_stage_depth {* buffer}`
	Note: This parameter is mutually exclusive with the `-path_length` parameter.
	Data_type: list, optional

`-through pin`	Specifies the pin to query delays through. This means that only paths from a source of the given skew group, through group, will be considered. *Data_type*: (pin|port), optional
`-to pin`	Specifies the pin to query delays to. This means that only paths from a source of the given skew group to this pin will *Data_type*: (pin|port), optional
`-virtual_delays_only`	Specifies that CTS will return the skew, shortest path delay, or longest path delay only considering the virtual delays will not include real delays added by clustering buffers and existing cells in the design. *Data_type*: bool, optional
`-wire_delays_only`	Specifies that CTS will return the skew, shortest path delay, or longest path delay only considering the wire delays in *Data_type*: bool, optional

Examples

- Use the following attribute to list all active skew groups in all clocks:
  ```
  get_db skew_groups
  m_clk/PM_HL_FUNC m_digit_clk/PM_HL_FUNC m_dsram_clk/PM_HL_FUNC m_ram_clk/PM_HL_FUNC m_rcc_clk/PM_HL_FUNC m_spi_clk/PM_HL_FUNC
  refclk/PM_HL_FUNC
  ```
- Use the following attribute to list the sinks in the specified skew group:
  ```
  cts_sinks -skew_group m_clk/PM_HL_FUNC
  TDSP_DS_CS_INST/g395/A1 ROM_512x16_0_INST/CLK SPI_INST/present_state_reg[0]/CP SPI_INST/dflag_reg/CP S..
  ```
- Use the following command to get the longest source to sink delay for the specified sink:
  ```
  get_skew_group_delay -skew_group m_clk/PM_HL_FUNC -to SPI_INST/dflag_reg/CP
  ```
- Use the following command to get the number of buffers and inverters along the specified path type in the skew group, `PM_HL_FUNC`:
  ```
  get_skew_group_delay -summarize_stage_depth {buffer inverter} -skew_group  m_dsram_clk/PM_HL_FUNC
  ```
  The software returns the following value:
  ```
  4
  ```

Related Information

- `create_skew_group`
- `delete_skew_groups`
- `get_db`
- `get_skew_group_path`
- `report_skew_groups`
- `set_db`
- `update_skew_group`
- The following sections in Clock Tree Synthesis chapter of *Innovus Stylus Common UI User Guide*
  - Reporting

- Manual Setup and Adjustment of the Clock Specification

gui_ctd_close

```
gui_ctd_close
[-help]
[-all | -id WindowIDName]
```

Closes open CTD windows. When no parameter is specified, the command closes the current active CTD window.

Parameters

`-help`	Outputs a brief description that includes type and default information for each gui_ctd_close parameter. For a detailed description of the command and all of its parameters, use the man command: `man gui_ctd_close`.
`-all`	Closes all open CTD windows. *Data_type*: bool, optional
`-id` *WindowIDName*	Closes the CTD window with the specified window ID. **Note**: The parameters, `-all` and `-id` are mutually exclusive. *Data_type*: string, optional

Examples

- The following command closes all CTD windows in the current session:
  ```
  gui_ctd_close -all
  ```
- The following command closes the CTD window with the ID, `user_id2`:
  ```
  gui_ctd_close -id user_id2
  ```

Related Information

- `gui_ctd_open`
- `gui_ctd_get_title`
- `gui_ctd_set_title`
- `gui_ctd_get_window_id`
- `gui_ctd_highlight`
- Clock Menu chapter of the *Innovus Stylus Common UI Menu Reference*.

gui_ctd_get_title

```
gui_ctd_get_title
[-help]
[-all | -id WindowIDName]
```

Retrieves the titles of the CTD windows. If no parameters are specified, the command retrieves the title of the current active CTD window.

Note: Before using this command, ensure that a CTD window is open. You can open a CTD window using the `gui_open_ctd` command.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `gui_ctd_get_title` parameter. For a detailed description of the command and all of its parameters, use the man command: `man gui_ctd_get_title`.
`-all`	Retrieves the IDs of all CTD windows. *Data_type*: bool, optional
`-id` *WindowIDName*	Retrieves the title of the CTD window with the specified ID. **Note**: The parameters, `-all` and `-id` are mutually exclusive. *Data_type*: string, optional

Examples

- The following command retrieves the titles of all CTD windows in the current session:

  ```
  gui_ctd_get_title -all
  ```

  {user_id1 : Clock Tree Debugger : title1} {user_id2 : Clock Tree Debugger : title2} {user_id3 : Clock Tree Debugger : title3}
- The following command retrieves the title of the CTD window with the ID, `user_id2`:

  ```
  gui_ctd_get_title -id user_id2
  ```

  ```
  user_id2 : Clock Tree Debugger : title2
  ```

  where, `user_id2` is the window ID, Clock Tree Debugger is the window type, and `title2` is the mapping title for this window.

Related Information

- `gui_ctd_open`
- `gui_ctd_close`
- `gui_ctd_set_title`
- `gui_ctd_get_window_id`
- `gui_ctd_highlight`
- Clock Menu chapter of the *Innovus Stylus Common UI Menu Reference*.

gui_ctd_get_window_id

```
gui_ctd_get_window_id
[-help]
[-all]
[-detail]
```

Retrieves the IDs of the CTD windows. If no parameters are specified, the command retrieves the window ID of the current active CTD window. For example, if you create a window with the ID, `user_id`, and then run this command without specifying any parameters, the software displays the following information:

```
gui_ctd_open -title my_title -id user_id
gui_ctd_get_window_id
user_id
```

Note: Before using this command, ensure that a CTD window is open. You can open a CTD window using the `gui_ctd_open` command.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `gui_ctd_get_window_id` parameter. For a detailed description of the command and all of its parameters, use the man command: `man gui_ctd_get_window_id`.
`-all`	Retrieves the IDs of all CTD windows. *Data_type*: bool, optional
`-detail`	Retrieves the IDs of CTD windows, with a mapping to their titles. *Data_type*: bool, optional

Examples

- The following command retrieves the ID of all CTD windows in the current session:

  ```
  gui_ctd_get_window_id -all

  user_id1 user_id2 user_id3
  ```

- The following command retrieves the IDs of CTD windows, with a mapping of their titles:

  ```
  gui_ctd_get_window_id -detail

  user_id2 : Clock Tree Debugger : title2
  ```

  where, `user_id2` is the window ID, Clock Tree Debugger is the window type, and `title2` is the mapping title for this window.

Related Information

- `gui_ctd_open`
- `gui_ctd_close`
- `gui_ctd_get_title`
- `gui_ctd_set_title`
- `gui_ctd_highlight`
- Clock Menu chapter of the *Innovus Stylus Common UI Menu Reference*.

gui_ctd_highlight

```
gui_ctd_highlight
[-help]
[-color color_name]
[-from clock_root]
[-index color_index]
[-out_file file_name]
[-to sink | -through instance_name]
```

Highlights the clock tree path from the clock root to the pin or from one pin to the other if the two pins are on the same path using the specified color or color index. The highlighted clock tree path can be viewed in the Clock Tree Viewer of the CCOpt Clock Tree Debugger (CTD). You can also use the Highlight Clock Path option in the context menu of the CTD to highlight the complete clock tree path of a selected pin to the clock root.

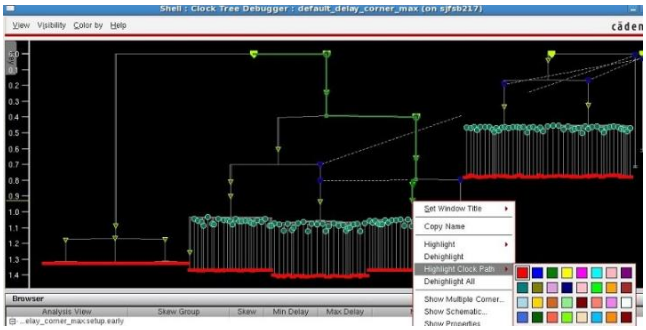
Parameters

`-help`	Outputs a brief description that includes type and default information for each `gui_ctd_highlight` parameter. For a detailed description of the command and all of its parameters, use the man command: `man gui_ctd_highlight`.
`-color color_name`	Specifies the color to be used for highlighting the clock tree path that is being traced. *Default*: red *Data_type*: string, optional
`-from clock_root`	Specifies the instance name as the starting point of the highlight. If this parameter is not specified, the starting point of the highlight is the clock root of the specified sink. *Data_type*: string, optional
`-index color_index`	Specifies the range of colors that can be used to highlight the clock tree path. Minimum is 1 and maximum is 32. The range corresponds to the colors in the color picker provided in the *Highlight Clock Path* option in the context menu of the Clock Tree Viewer. The number 1 means red, 2 means blue, and so on. An example is shown in the image below. *Default:* 1 or red. *Data_type*: int, optional, min=1, max=32

`-out_file file_name`	Specifies the name of the file in which the physical information for each instance and net in the clock path being highlighted will be dumped.
	The following information is saved in the file:
	`Name Cell Status Location Wire Length`
	`CLK_ROOT PAD Fixed (20,40)`
	`net_1 don't_touch 12.5`
	`Buf_1 BUFX1 placed (100,180)`
	`net_2 35`
	`Buf_2 BUFX2 placed (200, 120)`
	Name is the instance name or the net name
	Cell is the instance cell type. This information is only for instances
	Status is the place status for an instance and constrain status for a net. For dont_touch nets, the status is '`dont_touch`', otherwise this column is left blank
	Location is the physical location of the instance or the coordinate of that instance
	Wire Length is only for nets
	Data_type: string, optional

`-through instance_name`	Specifies the instance or pin through which the path should be traced. *Data_type*: string, optional
`-to sink`	Specifies the end point of the highlighted path. *Data_type*: string, optional

The image below shows the clock tree path highlighted in green, which corresponds to the color index 3. It also shows the color picker with the range of colors included in the color index.

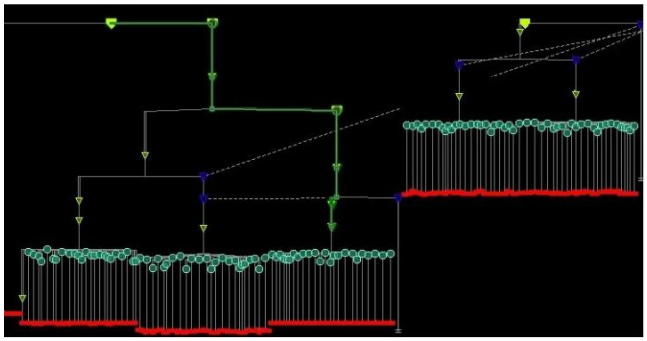


Example

The following command highlights the clock tree path for the specified pin, `ff1`, from the clock root in green color:

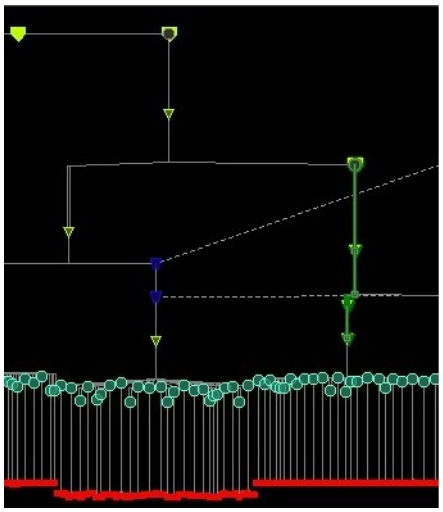
- `gui_ctd_highlight -to ff1 -index 3`

  The CTD shows the following:

- The following command highlights the path between two sinks that are on the same path, in green color:

  ```
  gui_ctd_highlight -to ff1  -from u1  -index 3
  ```

Related Information

- `gui_ctd_open`
- `gui_ctd_close`
- `gui_ctd_get_window_id`
- `gui_ctd_get_title`
- `gui_ctd_set_title`
- The following section in Clock Menu chapter of the *Innovus Stylus Common UI Menu Reference*
  - Context Menu of the Clock Tree Debugger

gui_ctd_open

```
gui_ctd_open
[-help]
[-id window_id_name]
[-include_reporting_only_skew_groups]
[-title ctd_window_id]
[-unit_delay]
```

Opens a Clock Tree Debugger (CTD) window with the user-defined window ID and title.

When run without specifying the ID and title, the command opens a CTD window with a system default window title. By default, the window title is the name of the current analysis corner, which shows in the CTD window. For example, "Clock Tree Debugger - func_rcbest".

By default, all CTD windows have their unique window ID. The mechanism of the default window ID is `ctdMain_0, ctdMain_1, ctdMain_2, ctdMain_3`....and so on. So, by default, the window ID of a new opened Clock Tree Debugger window is `ctdMain_0`.

The information in the title comprises the following:

```
Session title: Window type: Window title
```

where,

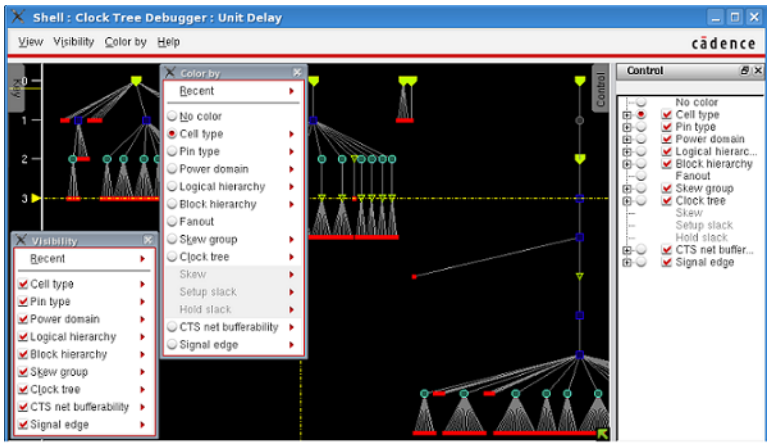
Window type is always “Clock Tree Debugger”.

If the user-specified session title is “`my_test.tcl`”, and the default primary analysis corner is “`primary_corner`”, then with `gui_ctd_open`, the total set of the window title line will be:

```
my_test.tcl : Clock Tree Debugger : primary_corner
```

Parameters

`-help`	Outputs a brief description that includes type and default information for each `gui_ctd_open` parameter. For a detailed description of the command and all of its parameters, use the man command: `man gui_ctd_open`.
`-id window_id_name`	Opens a Clock Tree Debugger window with a user-specified window ID. By default, the window IDs will be `ctdMain_0`, or `ctdMain_1`, and so on. *Data_type*: string, optional
`-include_reporting_only_skew_groups`	When this parameter is specified, the GUI includes the reporting-only skew groups. A reporting-only skew group is a skew group whose `cts_skew_group_constrains` attribute is set to `none`. Such a skew group imposes no clock balancing constraint and is not considered by CTS. By default, this setting if off, which means the reporting-only skew groups are omitted from the CTD. **Note**: Once the CTD is started, you cannot add or remove the reporting-only skew groups from those that are displayed. Start a new CTD window specifying the skew group filtering. *Data_type*: bool, optional
`-title ctd_window_id`	Specifies the user-defined title for the CTD window. Open a Clock Tree Debugger window with a user specified title. For example, if you specify `my_title` as the window title, when session title is `my_test.tcl` and the primary analysis corner is `primary_corner`, then the total set of the window title line will be: `my_test.tcl : Clock Tree Debugger : my_title` **Note**: The window’s ID if not specified using the `-id` parameter, will be system default ID. To view the default window ID, use the `gui_ctd_get_window_id` command. *Data_type*: string, optional

`-unit_delay`	Opens the CTD window in unit delay mode. The unit delay lets you view the depth of clock tree objects easily. This is useful when you have large designs with deep gate-level trees. When debugging clock trees, you can use this view to check how the clock trees are balanced by looking at the depth of the clock trees. Although, the unit delay mode can also be enabled using the *Unit delay* submenu in the *Visibility* menu of the CTD but this is only valid after running clock tree synthesis, and you still need to invoke timing calculation and extraction, which are time consuming. You can use this parameter to open the CTD in unit delay mode even if the design is not placed or without performing CTS. When the CTD is opened in this mode, the following options in CTD menus are disabled: *Visibility* menu: • Timing windows • Delays • Unit delay *Color by* menu and the *Control* panel: • Transition time • Total net cap • Max cap violation • Skew • Setup slack • Hold slack • Timing windows This is shown in below image.  *Data_type*: bool, optional

Examples

- The following command creates a CTD window ID with the default name, `ctdMain_0`:

  ```
  gui_ctd_open
  ```

  When you use the `gui_ctd_get_window_id` command to list the window ID, you get the above window ID:

  ```
  gui_ctd_get_window_id
  ctdMain_0
  ```

- The following command creates a CTD window ID with the user-specified name, `my_test`:

  ```
  gui_ctd_open -id my_test
  ```

  Now, when you use the `gui_ctd_get_window_id` command to list all window IDs, you get the above window IDs in the list:

```
gui_ctd_get_window_id -all
ctdMain_0 my_test
```

If you use the `gui_ctd_open` command with an existing window ID, it will pop up the corresponding CTD windows instead of creating a new one. This command will open the existing window, `my_test`.

```
gui_ctd_open -id my_test
```

- The following command creates a CTD window ID, `my_test` and CTD window title of `myTest.tcl`:

```
gui_ctd_open -id my_test -title myTest.tcl
```

When you use the `-detail` parameter of the `get_ctd_win_id` command, the following information is returned:

```
gui_ctd_get_window_id -detail
my_test "Clock Tree Debugger:myTest.tcl: func_rcbest"
```

where, `my_test` is the specified window ID, window type is Clock Tree Debugger, window title is `myTest.tcl`, and `func_rcbest` is the name of the current analysis corner.

Related Information

- `gui_ctd_close`
- `gui_ctd_get_title`
- `gui_ctd_set_title`
- `gui_ctd_get_window_id`
- `gui_ctd_highlight`
- Clock Menu chapter of the *Innovus Stylus Common UI Menu Reference*.

gui_ctd_write_histogram

```
gui_ctd_write_histogram
[-help]
[-csv]
-file fileName
[-id windowID]
```

Saves the CTD histogram to a file. By default, the file type is PNG. You can also specify that the file be saved in the CSV format.

Before using this command to save the histogram view, first open and configure the histogram in the GUI, including skew group and delay corners.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `gui_ctd_write_histogram` parameter. For a detailed description of the command and all of its parameters, use the man command: man `gui_ctd_write_histogram`.
`-csv`	Saves the histogram in CSV format. This parameter is optional. By default, the histogram is saved as a PNG file.
`-file` *fileName*	Specifies the name of the file where you want to save the histogram. This parameter is required.
`-id` *windowIDName*	Specifies the ID of the active CTD window. The default value is the ID of the current active window.

Example

The following command saves the histogram in the `histogram1.csv` file.

```
gui_ctd_write_histogram -file histogram1 -csv
```

Related Information

- `gui_ctd_open`
- `gui_ctd_get_title`
- `gui_ctd_set_title`
- `gui_ctd_get_window_id`
- `gui_ctd_highlight`
- `gui_ctd_write_view`
- Clock Menu chapter of the *Innovus Stylus Common UI Menu Reference*.

gui_ctd_write_view

```
gui_ctd_write_view
[-help]
-file fileName
[-id windowID]
[-legend_on]
[-type {PNG | GIF}]
```

Saves the current snapshot of the clock tree viewer into a file.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `gui_ctd_write_view` parameter. For a detailed description of the command and all of its parameters, use the man command: man `gui_ctd_write_view`.
`-file` *`fileName`*	Specifies the name of the file where you want to dump the display image. This parameter is required.
`-id` *`windowIDName`*	Specifies the window ID whose display is to be saved. The default value is the returned value of command `gui_ctd_get_window_id`.
`-legend_on`	Saves the whole window including the panels that are open.
`-type {PNG | GIF}`	Specifies whether the format of the image should be GIF or PNG. The default format is PNG.

Example

The following command saves the view snapshot in the `snapshot1.png` file.

```
gui_ctd_write_view -file snapshot1 -type PNG
```

Related Information

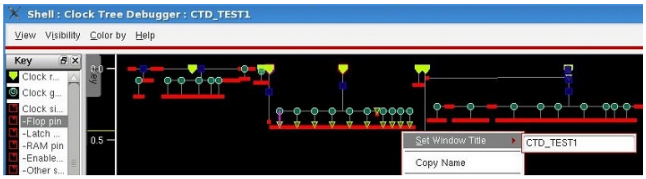
- `gui_ctd_open`
- `gui_ctd_get_title`
- `gui_ctd_set_title`
- `gui_ctd_get_window_id`
- `gui_ctd_highlight`
- Clock Menu chapter of the *Innovus Stylus Common UI Menu Reference*.

gui_ctd_set_title

```
gui_ctd_set_title
[-help]
title
[-id window_id_name]
```

Specifies the title of the CTD window. By default, this command is used to set the title of the current active CTD window either if a unique ID has not been specified for this window or if you want to change the existing window ID.

You can also set the window title from the context menu of the Clock Tree Viewer in the CTD. Click Set Window Title, type the title for the CTD session, and press enter. The new title appears in the top bar of the CTD. This is shown in below image.



Parameters

`-help`	Outputs a brief description that includes type and default information for each `gui_ctd_set_title` parameter. For a detailed description of the command and all of its parameters, use the man command: `man gui_ctd_set_title`.
`-id` *`window_id_name`*	
	Specifies the ID of the window whose title is to be set. If there are several CTD windows open in the current session, with window IDs, `user_id1, user_id3, user_id3`, and so on, with corresponding window titles, `title1, title2, title3`, and so on, then the following command will only change the `title1` of window with ID, `user_id1`, to `my_new_title`. `gui_ctd_set_title my_new_title -id user_id1` *Data_type*: string, optional
`title`	Specifies the new window title to be given to the specified window. This parameter is required. *Data_type*: string, required

Examples

- The following command sets the window title of the current CTD window, `my_title`, to `my_new_title`:

  ```
  gui_ctd_set_title my_new_title
  ```

  The active CTD window title will be changed from:

  ```
  my_test.tcl : Clock Tree Debugger : my_title
  ```

  to:

  ```
  my_test.tcl : Clock Tree Debugger : my_new_title
  ```

- The following command retrieves the window IDs of CTD windows, with a mapping of their window titles:

  ```
  gui_ctd_get_window_id -detail
  ```

  ```
  user_id2 : Clock Tree Debugger : title2
  ```

where, `user_id2` is the window ID, Clock Tree Debugger is the window type, and title2 is the mapping title for this window.

Related Information

- `gui_ctd_open`
- `gui_ctd_close`
- `gui_ctd_get_window_id`
- `gui_ctd_get_title`
- `gui_ctd_highlight`
- Clock Menu chapter of the *Innovus Stylus Common UI Menu Reference*.

gui_ctd_zoom

```
gui_ctd_zoom
[-help]
[-id string]
[-in | -out | -selected]
[-vertical_range {ymin ymax}]
```

Zooms into or out of the CTD window.

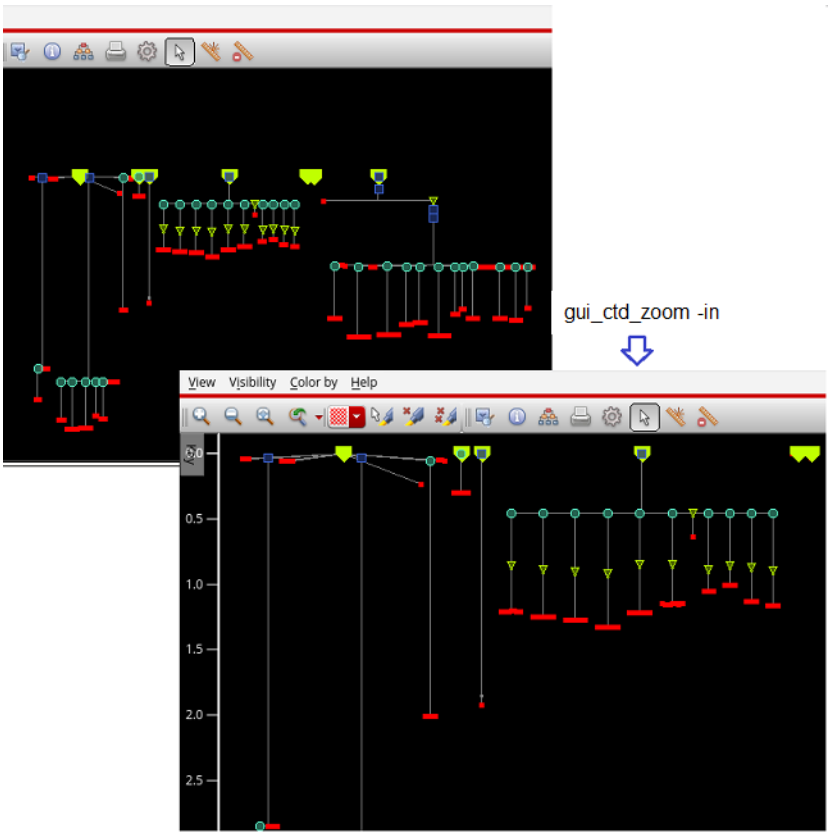
Parameters

`-help`	Prints a brief description of the `gui_ctd_zoom` command. For a detailed description of the command, use the `man` command: `man gui_ctd_zoom`.
`-id string`	Specifies the window ID of the CTD window in which to zoom in or zoom out. *Data_type*: string, optional
`-in`	Zoom into the CTD window. *Data_type*: bool, optional
`-out`	Zoom out of the CTD window. *Data_type*: bool, optional
`-selected`	Zoom to enclose all selected objects. *Data_type*: bool, optional
`-vertical_range {ymin ymax}`	Zoom to a specific vertical latency range. If CTD is in timing-based mode, the window will zoom to the clock tree parts where latency is between *ymin* and *ymax*, and the unit is the same as global timing unit. If CTD is in unit delay mode, the window will zoom to the clock tree parts where stage depth is between *ymin* and *ymax*. For example, if CTD is in timing-based mode, then the command, `gui_ctd_zoom -vertical_range {0 1.5}` will zoom to the clock tree parts where latency is between 0 and 1.5, the unit is the same as `get_time_unit`. If CTD is in unit delay mode, then the command, `gui_ctd_zoom -vertical_range {0 1.5}` will zoom to the clock tree parts where stage depth is between 0 and 1.5. **Note**: The window only zooms in vertical.

Examples

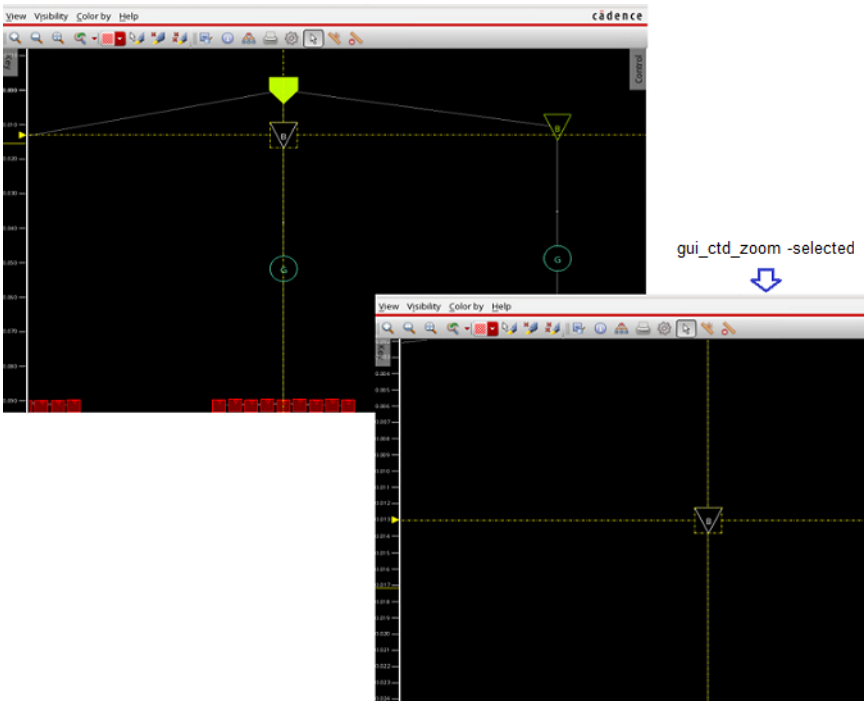
Use the following command to zoom in to the CTD:

```
gui_ctd_zoom -in
```



Select an instance and use the following command to zoom into the selected instance:

```
gui_ctd_zoom -selected
```



Related Information

- `gui_ctd_open`
- `gui_ctd_close`
- `gui_ctd_highlight`
- `gui_ctd_get_window_id`
- `gui_ctd_get_title`
- `gui_ctd_set_title`
- Clock Menu chapter of the *Innovus Stylus Common UI Menu Reference*.

merge_clock_tree_instances

```
merge_clock_tree_instances
[-help]
[-only_above_flops]
[-only_clock_gates]
[-only_clock_logics]
```

When specified without any parameters, the command merges both clock gates and clock logic. You can specify the parameters, `-only_clock_gates` or `-only_clock_logic` to merge only clock gates or clock logics, respectively. The command ignores the settings of the `cts_merge_clock_logic and cts_merge_clock_gates` attributes.

The log will show the names of instances that will be merged by saying, “Merging clock gates...”, or "Merging clock logics", or ”Merging clock gates and clock logics". The summary tables in the log will list the number of instances that are merged. This is shown in the "Examples" section below.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `merge_clock_tree_instances` parameter. For a detailed description of the command and all of its parameters, use the man command: `man merge_clock_tree_instances`.
`-only_above_flops`	
	Specifies that CCOpt should merge clock gates that only drive flops and not the ones that drive RAMs, latches, black boxes, or any other clock tree sink that is not a flop. *Data_type*: bool, optional
`-only_clock_gates`	
	Specifies that only clock gates should be merged and not clock logics. *Data_type*: bool, optional
`-only_clock_logics`	
	Specifies that only clock logics should be merged and not clock gates. *Data_type*: bool, optional

Examples

- When you run the following command, the summary table that is generated is shown below:
  `merge_clock_tree_instances -only_clock_gates`

  Summary of the merge of duplicate siblings

```
------------------------------------------------------------
Description                          Number of occurrences
------------------------------------------------------------
Total clock gates                              2
Globally unique enables                        1
Potentially mergeable clock gates              1
Actually merged clock gates                    0
------------------------------------------------------------

------------------------------------------------------
Description                       Number of occurrences
------------------------------------------------------
Total clock logics                         2
Actually merged clock logics               0
------------------------------------------------------

------------------------------------------------------------
Cannot merge reason                  Number of occurrences
------------------------------------------------------------
UniqueUnderParent                              1
ClockLogicMergingDisabledOnTree                2
------------------------------------------------------------
```

  In the above example, the first table is for clock gates – there are two in this design. No clock gates were merged.
  The second table is for clock logics – there are two in this design. No clock gates were merged.
  The third table provides the reasons as to why the instances could not be merged. Logic merging has been disabled because of the `-only_clock_gates` parameter. If the logics cannot be merged, then the clock gates are not siblings and cannot be merged.

- When you run the following command, the summary table that is generated is shown below.
  `merge_clock_tree_instances`

```
Summary of the merge of duplicate siblings
------------------------------------------------------
Description                     Number of occurrences
------------------------------------------------------
Total clock gates                        2
Globally unique enables                  1
Potentially mergeable clock gates        1
Actually merged clock gates              1
------------------------------------------------------

--------------------------------------------------
Description                    Number of occurrences
--------------------------------------------------
Total clock logics                      2
Actually merged clock logics            1
--------------------------------------------------

---------------------------------------------
Cannot merge reason     Number of occurrences
---------------------------------------------
UniqueUnderParent                1
---------------------------------------------
```

  In the above example, the clock logics got merged - one logic got merged with its sibling logic. This means that the clock gates became siblings and were merged.

Related Information

- The following attributes:
  - `cts_clock_gating_opt_only_above_flops`
  - `cts_cannot_merge_reasons`

For more information about the above attributes, see the cts Category Attributes chapter.

place_detail_clock_tree

```
place_detail_clock_tree
[-help]
```

Performs clock placement refinement with cell halos. Use this command at the end of CTS to ensure that all clock cells are placed legally and obey cell halo and density constraints. You can use `report_clock_tree_halo_violations` to report any clock halo constraint violations. Then run `place_detail_clock_tree` to fix any reported violations. You can re-run the `report_clock_tree_halo_violations` command to verify that these instances have been fixed.

Parameter

`-help`	Outputs a brief description for the `place_detail_clock_tree` command parameter. For a detailed description of the command, use the man command: `man place_detail_clock_tree`.

Example

The following set of commands show that halo violations are reported with `report_clock_tree_halo_violations` command, then the `place_detail_clock_tree` command performs clock placement refinement with cell halos and then `report_clock_tree_halo_violations` is re-run to verify that the instances have been fixed:

```
report_clock_tree_halo_violations

Clock Cell Halo Rule Check in Summary
=====================================

-----------------------------------------------------
Clock Instances with cell_halo Failed instances
-----------------------------------------------------
clock 6 3
-----------------------------------------------------

Total number of violating instances: 3

Clock Cell Halo Rule Violations
===============================

-------------------------------------------------------------------------------------------------
No. Cell Instance In clock Cell Halo (x, y) Violating instance From clock Intrusion (x, y)
-------------------------------------------------------------------------------------------------
1 CLKBUFX12 CTS_ccl_BUF_clock_G0_L1_1 clock (15, 0) CTS_ccd_BUF_clock_G1_L2_5 clock (3.22, 3.69)
2 CLKBUFX12 CTS_ccl_BUF_clock_G0_L2_6 clock (15, 0) CTS_cdb_BUF_clock_G0_L3_4 clock (2.3, 3.69)
3 CLKBUFX12 CTS_csk_BUF_clock_G1_L2_1 clock (15, 0) CTS_ccd_BUF_clock_G1_L3_5 clock (2.12, 3.69)
-------------------------------------------------------------------------------------------------

place_detail_clock_tree

Calling refine place with halo constraints.
Have found 0 CTS cell-density overrides:
Refine Place commencing.
Leaving CCOpt scope...
*
* Starting clock placement refinement...
*
* First pass: Refine non-clock instances...
*
*** Starting place_detail (0:00:21.6 mem=897.0M) ***
Total net bbox length = 1.613e+04 (9.844e+03 6.288e+03) (ext = 6.967e+01)
Density distribution unevenness ratio = 14.001%
Move report: Detail placement moves 0 insts, mean move: 0.00 um, max move: 0.00 um
Runtime: CPU: 0:00:00.0 REAL: 0:00:00.0 MEM: 899.0MB
Summary Report:
Instances move: 0 (out of 255 movable)
Mean displacement: 0.00 um
Max displacement: 0.00 um
Total net bbox length = 1.613e+04 (9.844e+03 6.288e+03) (ext = 6.967e+01)
Runtime: CPU: 0:00:00.0 REAL: 0:00:00.0 MEM: 899.0MB
*** Finished place_detail (0:00:21.6 mem=899.0M) ***
*
```

```
* Second pass: Refine clock instances...
*
*** Starting place_detail (0:00:21.9 mem=899.0M) ***
Total net bbox length = 1.613e+04 (9.844e+03 6.288e+03) (ext = 6.967e+01)
Density distribution unevenness ratio = 0.000%
Move report: Detail placement moves 28 insts, mean move: 4.75 um, max move: 11.98 um
Max move on inst (CTS_ccd_BUF_clock_G1_L2_5): (73.14, 29.52) --> (68.54, 22.14)
Runtime: CPU: 0:00:00.0 REAL: 0:00:00.0 MEM: 899.0MB
Summary Report:
Instances move: 28 (out of 333 movable)
Mean displacement: 4.75 um
Max displacement: 11.98 um (Instance: CTS_ccd_BUF_clock_G1_L2_5) (73.14, 29.52) -> (68.54, 22.14)
Length: 7 sites, height: 1 rows, site name: tsm12site, cell type: CLKBUFX8
Total net bbox length = 1.621e+04 (9.903e+03 6.304e+03) (ext = 6.967e+01)
Runtime: CPU: 0:00:00.0 REAL: 0:00:00.0 MEM: 899.0MB

*** Finished place_detail (0:00:21.9 mem=899.0M) ***
*
* Moved 7 and flipped 4 of 78 clock instance(s) during refinement.
* The largest move was 11.98 microns for CTS_ccd_BUF_clock_G1_L2_5.
*
* Finished with clock placement refinement.
*

Leaving CCOpt scope done.
Refine Place finished.

Summary of Movements
Total moves = 6
Max Move : inst : CTS_ccd_BUF_clock_G1_L2_5 has moved from ( 73.140um , 29.520um )
to ( 68.540um , 22.140um ) distance = 11.98um.
Average Move = 6.6
6
report_clock_tree_halo_violations

Clock Cell Halo Rule Check in Summary
=====================================

-----------------------------------------------------
Clock Instances with cell_halo Failed instances
-----------------------------------------------------
clock 6 0
-----------------------------------------------------

Total number of violating instances: 0

Clock Cell Halo Rule Violations
==============================

-------------------------------------------------------------------------------------------
No. Cell Instance In clock Cell Halo (x, y) Violating instance From clock Intrusion (x, y)
-------------------------------------------------------------------------------------------
(empty table)
-------------------------------------------------------------------------------------------
```

Related Information

- report_clock_tree_halo_violations
- The following sections in Clock Tree Synthesis chapter of *Innovus Stylus Common UI User Guide*
  - Halo Violations
  - Cell Halos

read_cts_config

```
read_cts_config
[-help]
input_path
```

Reads the CTS configuration from the specified directory.

Note: The CTS configuration will not be read when `read_db -mmmc_file` option is set. In this case, the software issues the following warning:

```
**WARN: (IMPIMEX-4016): The saved CCOpt configuration will not be restored, as the -mmmc_file option has been set. To restore the CCOpt configuration try restoring the design again without using the -mmmc_file option.
```

Parameter

`-help`	Outputs a brief description of the `read_cts_config` parameter.
input_path	Specifies the directory path from which to read the CTS configuration. *Data_type*: string, required

Example

- The following command reads the CTS configuration from the specified path:
  ```
  read_cts_config ../ccopt_tutorial/work
  ```

Related Commands

- `check_cts_config`
- `reset_cts_config`
- `write_cts_config`

report_cts_cell_name_info

```
report_cts_cell_name_info
[-help]
```

Returns the list of cell name codes. All new cell instances created by CTS have an identifying code in their name to let you determine why that cell was created. This command lists all available codes and their meanings.

Parameters

`-help`	Outputs a brief description for the `report_cts_cell_name_info` parameter. For a detailed description of the command, use the man command: `man report_cts_cell_name_info`.

Example

- The following command returns a list of all available codes and their meanings:

```
report_cts_cell_name_info

Creators:
CTSG Unnamed - this should not be used any more.
cuk Cts: Unknown creator, will not appear in the netlist.
ccl_a Cts: Created during clustering by the agglom clustering algorithm.
cbi Cts: Created by the swapping buffers and inverters for power.
cex Cts: Existing cells in the clock tree which cannot be removed.
coi Cts: Cells created as a result of cancelling out inversions
lbl Cts: Created by the clustering process to meet the slew target
ccl Cts: Created by the clustering process to meet the slew target
ccd Cts: Created by clustering for balancing the tree - these cells are not necessary to meet the slew target.
cci Cts: Created by the clustering process to fix inversion.
csf Cts: Created by the CTS slew fixing step in cases where clustering did not meet the slew target.
cms Cts: Created during the process of physically moving clock gates to improve their enable timing.
cid Cts: Created by CTS on the outputs of weak driving cells to reduce insertion delay.
cdb Cts: Created by CTS to balance the delays in the clock tree.
cdbw Cts: Created by CTS to balance the delays in the clock tree.
cwb Cts: Created by CTS to balance the wire delays in the clock tree.
cfo Cts: Created by CTS to reduce fanout skew.
csk Cts: Created by the CTS skew fixing step to finely balance the clock tree.
cmf Cts: Created by CTS to buffer long nets.
cbc Cts: Created by the clock tree conditioning step to clone off sub trees that cannot optimized by gated synthesis, such as those
parts above RAMs, black boxes, and lockup latches.
css Cts: Created by the early cloning of simple sink allocations.
cdc Cts: A clock driver created by adding driver cell process for attribute cts_add_driver_cell.
cpd Cts: A clock driver created below an input port or above an output port specified by attribute cts_add_port_driver.
ccg Cts: A clock gate created by one of the gated synthesis algorithms.
cse Cts: A clock driver created above exclude pins to remove them from the clock tree.
cfh Cts: A clock driver created as part of a flexible H-tree.
cat Cts: Created by the add_clock_tree_source_group_roots command
cpc_drv Cts: A clock driver created by post conditioning.
cpc_sk Cts: A clock driver created by post conditioning.
PRO Post-route optimization.
PRO_drv Post-route optimization.
PRO_sk Post-route optimization.
sfc Cloned by slew fixing.
ccc Cts: Created by the clone_clock_cells command.
grb Cts: A clock driver created by global route buffering.
sgb Cts: A clock driver created by source group balancing.
idc Cloned to reduce insertion delay
vgb Cts: A clock driver created by VG buffering.
USK skewClock
cff Cts: A clock driver created by fanout fixing.
```

report_ccopt_worst_chain

```
report_ccopt_worst_chain
[-help]
[-check_type {setup | hold}]
[-out_file filename]
[-through pin | -through_file filename]
[-view viewname]
```

This command displays information about the worst chain, which is the chain that contains the path with the worst negative slack or WNS in the design. CCOpt is able to move slack between paths, so the fundamental timing constraint is not from one flop to the next, but chains of flop–to–flop paths. At each flop, timing can be shifted before or after the flop in order to make the physical optimization problem easier.

The command reports a chain as a series of nodes linked together by timing paths. Additionally, it contains information to let you determine how well the chain has been optimized. This report contains the following information:

- The timing chain containing the worst flop-to-flop path, which is identified in the report output by the string, "*WNS*"
- Timing path and constraint information to show why that WNS cannot be improved

Parameters

`-help`	Outputs a brief description that includes type and default information for each `report_ccopt_worst_chain` parameter. For a detailed description of the command and all of its parameters, use the man command: `man report_ccopt_worst_chain`.
`-check_type setup | hold`	
	Specifies whether setup views or hold views are used for the report. *Default*: By default, only the worst chain for setup views is reported. *Data_type*: enum, optional
`-out_file filename`	Specifies that a copy of the contents of the report will be saved to the specified file name. *Data_type*: string, optional
`-through pin`	Specifies the path through which the worst chain is being reported. When this parameter is specified, the chain containing the given pin or port is reported. The timing path which contains the `-through` pin is labelled with "*SEL*". When this parameter is not specified, the chain with the WNS timing path is reported. The WNS timing path in the chain is labelled "*WNS*". *Data_type*: (pin|port), optional
`-through_file filename`	
Creates a worst chain report for each pin or port in the specified file. **Note**: The parameters, `-through` and `-through_file` are mutually exclusive. *Data_type*: string, optional	
`-view viewname`	Specifies the view for which the worst chain is being reported. *Data_type*: (analysis_view), optional

Example

Following is an example of the report output:

```
report_ccopt_worst_chain
```

```
,-o PM_INST/power_switch_enable_reg_reg
| |     .../CP @(unknown) constraint=(unknown) chosen=(unknown)
| |           location=(209.510,145.600) slew=(launch: 0.000ns, capture: 0.000ns)
| |   slack 1.036ns .../Q -> .../D (distance: 42.280um)
| |                delays=(launch: 0.000ns, datapath: 0.964ns, capture: 0.000ns, adjust: 2.000ns)
| |                launch/capture clock m_clk in analysis view AV_HL_FUNC_MAX_RC1
| |                path group reg2reg
`-o PM_INST/power_switch_enable_reg_reg
  |     .../CP @(unknown) constraint=(unknown) chosen=(unknown)
  |           location=(209.510,145.600) slew=(launch: 0.000ns, capture: 0.000ns)
  |   slack -2.476ns .../Q -> .../NRESTORE (distance: 314.740um)
  |                 delays=(launch: 0.000ns, datapath: 4.476ns, capture: 0.000ns, adjust: 2.000ns)
  |                 launch clock m_clk in analysis view AV_HL_FUNC_MAX_RC1
  |                 capture clock refclk in analysis view AV_HL_FUNC_MAX_RC1
  |                 path group reg2reg
  o TDSP_CORE_INST/DECODE_INST/ir_reg[8]
  |     .../CP @(unknown) constraint=(unknown) chosen=(unknown)
  |           location=(393.070,150.640) slew=(launch: 0.000ns, capture: 0.000ns)
  |   *WNS* -14.670ns .../Q -> .../D (distance: 454.300um)
  |                  delays=(launch: 1.000ns, datapath: 16.670ns, capture: 1.000ns, adjust: 2.000ns)
  |                  launch/capture clock refclk in analysis view AV_HL_FUNC_MAX_RC1
  |                  path group reg2reg
,-o TDSP_CORE_INST/EXECUTE_INST/acc_reg[7]
| |     .../CP @(unknown) constraint=(unknown) chosen=(unknown)
| |           location=(328.810,125.440) slew=(launch: 0.000ns, capture: 0.000ns)
| |   slack -11.200ns .../Q -> .../D (distance: 334.320um)
| |                  delays=(launch: 1.000ns, datapath: 16.670ns, capture: 1.000ns, adjust: 2.000ns)
| |                  launch/capture clock refclk in analysis view AV_HL_FUNC_MAX_RC1
| |                  path group reg2reg
`-o TDSP_CORE_INST/EXECUTE_INST/acc_reg[7]
```

Related Information

- ccopt_design
- create_clock_tree
- create_clock_tree_spec
- The following sections in Clock Tree Synthesis chapter of *Innovus Stylus Common UI User Guide*
  - Worst Chain
  - Chains

report_clock_tree_convergence

```
report_clock_tree_convergence
[-help]
[-out_file file_name]
```

This command is used to report a summary of the convergence above clock sinks and to identify the names of sinks with the highest number of clock path networks. If there are sinks with many paths leading to them, there can be a considerable impact on runtime. Reducing the complexity of the logic, or excluding sinks that need not have timing constraints from the clock trees can help improve runtime. An example of the report is shown below.

```
Convergence above clock sinks
=============================
-------------------------------------------------------------------
Number of paths to sink PIOs DFFs  Other sinks Total
-------------------------------------------------------------------
1                       1    10118 0           10119
8                       54   2134  183         2371
16                      13   311   18          342
32                      2    0     0           2
700                     3    164   5           172
1400                    5    0     0           5
-------------------------------------------------------------------
Sinks with most paths leading to them
=====================================
---------------------------------
Sink             Number of paths
---------------------------------
Owest/nout[0]    1400
Owest/nout[1]    1400
Owest/nout[2]    1400
Owest/nout[3]    1400
Owest/nout[4]    1400
---------------------------------
---------------------------------
```

In the above sample report, 5 primary IOs have 1400 paths leading to them from the clock sources.The report also shows a table with the names of sinks with most paths leading to them. In the "Sinks with most paths leading to them" section of the report, the five sinks, `Owest/nout[n]`, have 1400 paths leading to them. These sinks are likely to be problematic and need to be checked.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `report_ccopt_clock_tree_convergence` parameter. For a detailed description of the command and all of its parameters, use the man command: `man report_clock_tree_convergence`.
`-out_file` *`file_name`*	Specifies the file name in which the report will be saved. *Data_type*: string, optional

Related Information

- `create_clock_tree_spec`
- `check_clock_tree_convergence`
- The following sections in Clock Tree Synthesis chapter of *Innovus Stylus Common UI User Guide*
  - Clock Tree Convergence
  - Reporting

report_clock_tree_halo_violations

```
report_clock_tree_halo_violations
[-help]
[-add_markers | -clear_markers]
[-out_file file_name]
[-max_violation_count number]
[-summary]
```

This command is used to report clock instances with clock halo violations.

By default, when the command is specified without any arguments, it will check for halo violations for clock halos defined using all of the following cts category attribute pairs:

- `cts_cell_halo_x` / `cts_cell_halo_y`
- `cts_cell_density` / `cts_adjacent_rows_legal`
- `cts_cell_halo_sites` / `cts_cell_halo_rows`

The report can be viewed as a text file and in the Violation Browser within the Stylus GUI.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `report_clock_tree_halo_violations` parameter. For a detailed description of the command and all of its parameters, use the man command: `man report_clock_tree_halo_violations`.
`-add_markers`	Adds any detected clock halo violations to the Violation Browser. *Data_type*: bool, optional
`-clear_markers`	Removes any markers added to the Violation Browser by the `report_clock_tree_halo_violations` command. *Data_type*: bool, optional
`-max_violation_count` *number*	Specifies the number of violations to report for each clock. *Data_type*: int, optional
`-out_file` *file_name*	Specifies the name of the file in which the contents of the report will be saved. *Data_type*: string, optional
`-summary`	Prints a report providing information about the instances with cell halos and the instances with cell halo violations or "Failed instances". This report contains a summary table titled, "Clock Cell Halo Rule Check in Summary". *Data_type*: bool, optional

Example

The following command prints the summary of the cell halo violation report:

```
report_clock_tree_halo_violations -summary
```

The software prints the following report:

```
Clock Cell Halo rule Check in Summary
=====================================

-----------------------------------------------------------
Clock       Instances with cell_halo Failed Instances
-----------------------------------------------------------
funcclk1           4                   3
funcclk2           4                   3
funcclk3           4                   2
-----------------------------------------------------------
total number of violating instances: 8
```

Use the following command to view the detailed "Clock Cell Halo Rule Violations" table.

```
report_ccopt_cell_halo_violations
```

The software prints the following report:

```
Clock Cell Halo Rule Violations
===============================

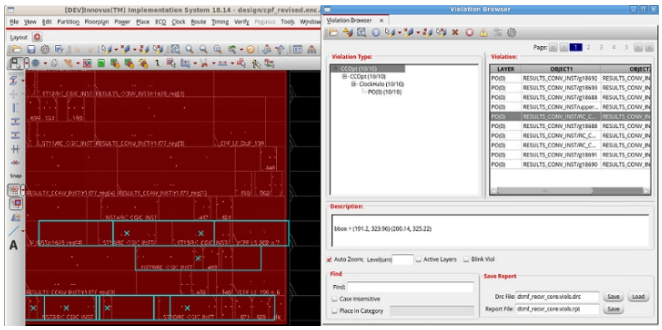
-------------------------------------------------------------------------------------------------------------------------------------------------------------
No. Cell      Instance                          In clock   Power domain  Cell Halo (x, y)   Violating instance                                  From clock    Intrusion (x, y)
-------------------------------------------------------------------------------------------------------------------------------------------------------------
 1  INVD1BWP  RESULTS_CONV_INST/g18692          m_rcc_clk  TD8PCore      (3, 0)             RESULTS_CONV_INST/RC_CG_HIER_INST8/RC_CGIC_INST  m_rcc_clk     (2.94, 1.26)
 2  INVD1BWP  RESULTS_CONV_INST/g18693          m_rcc_clk  TD8PCore      (3, 0)             RESULTS_CONV_INST/RC_CG_HIER_INST9/RC_CGIC_INST  m_rcc_clk     (2.94, 1.26)
 3  INVD1BWP  RESULTS_CONV_INST/g18688          m_rcc_clk  TD8PCore      (3, 0)             RESULTS_CONV_INST/RC_CG_HIER_INST8/RC_CGIC_INST  m_rcc_clk     (2.58, 1.26)
```

Note: The "Power Domain" column helps to understand where the halo value comes from in the case where the halo has been set per power domain.

Use the following set of commands to view the clock halo violations in the Violation Browser

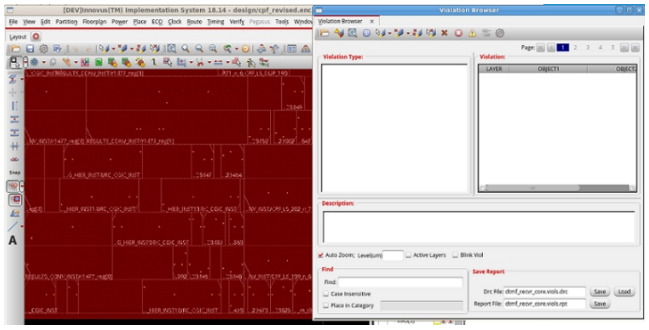
- Create cell halo violations using the following attribute:
  `set_db cts_cell_halo_x 3`
- Use the following command to add the detected clock halo violations to the Violation Browser and display them in the GUI:
  `report_clock_tree_halo_violations -add_markers`

  You will see the following:



- Use the following command to clear the clock halo violations in the Violation Browser and delete markers in the GUI:
  `report_clock_tree_halo_violations -clear_markers`

  You will see the following:



Related Topics

- The following sections in Clock Tree Synthesis chapter of *Innovus Stylus Common UI User Guide*
  - Halo Violations
  - Cell Halos
- The following attributes:
  - `cts_cell_halo_sites`
  - `cts_cell_halo_rows`
  - `cts_cell_halo_x`

- `cts_cell_halo_y`

For more information, see cts Category Attributes chapter in the *Innovus Stylus Common UI Text Command Reference*

report_clock_tree_structure

```
report_clock_tree_structure
[-help]
[-check_type {setup | hold}]
[-clock_trees {list_of_clock_trees}]
[-delay_corner corner]
[-delay_type {early | late}]
[-expand_below_logic]
[-expand_generated_clock_trees {independently | inline | independently_and_inline}]
[-include_reporting_only_skew_groups]
[-out_file filename]
[-show_sinks]
[-update_timing]
```

Reports the structure of the clock network as a text report. When the command is run without specfying the `-update_timing` parameter, the report format is as follows:

- The summary information is added below the name of the clock tree
- The slew, capacitance, and co-ordinate information is not included
- The input and output pin information is merged in a single line

For details, see the examples below.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `report_clock_tree_structure` parameter. For a detailed description of the command and all of its parameters, use the man command: `man report_clock_tree_structure`.
`-check_type {setup | hold}`	
	Specifies whether setup or hold views are used for the report. By default, setup view is used for the report. *Data_type*: enum, optional
`-clock_trees {list_of_clock_trees}`	
	Restricts the report to the specified clock trees. *Data_type*: list, optional
`-delay_corner`	Reports in the specified delay corner. By default, the software will use the primary-half delay corner that is set using the CCOpt attribute, `cts_primary_delay_corner`. *Data_type*: (delay_corner), optional
`-delay_type {early | late}`	
	Reports in the specified delay type, early or late. When late is specified, only the late half of the delay corners are reported. When early is specified, only the early half of the delay corners are reported. By default, late half of the delay corners are reported. *Data_type*: enum, optional
`-expand_below_logic`	

	When this parameter is specified, the subtree below each multi-input logic will be printed in its entirety every time it is encountered. When this parameter is not specified, then by default, the clock tree below a multi-input clock logic is reported only once, which is at the first place it is encountered in the report. Subsequent visitations of the subtree will simply note the presence of an omitted subtree, for example, "...71 `fanout omitted`". **Note**: For clock networks with ladders of reconverging clock logic, this can produce large volumes of output. *Data_type*: bool, optional
`-expand_generated_clock_trees {independently | inline | independently_and_inline}`	
	Specifies how generated clock trees should be displayed. The following options are available: `independently`: display generated clock trees at the top level. This is the default display. `inline` : display generated clock trees as fanout of their generator inputs `independently_and_inline`: display generated clock trees both at the top level and as fanout of their generator inputs *Data_type*: enum, optional
`-include_reporting_only_skew_groups`	When specified, the report additionally includes the reporting-only skew groups. A reporting-only skew group is a skew group whose `cts_skew_group_constrains` attribute is set to `none`. Such a skew group imposes no clock balancing constraint and is not considered by CTS. By default, the reporting-only skew groups are not included in the report. *Data_type*: bool, optional
`-out_file` *filename*	Saves a copy of the report in the specified file. *Default*: `{}` *Data_type*: string, optional
`-show_sinks`	Includes all clock network sinks in the report. This parameter controls whether the clock network sinks, which are the non-generator flops/latches, are included in the report or not. By default, these are not included in the interest of minimizing the report output. Instead, a count of the number of sinks is displayed,for example. `"... 209 sinks omitted"`. When this parameter is specified, per-skew group latencies for each sink are included. **Note**: The report identifies macros in the clock tree. This means that if your clock sink is a macro, the report will show “macro sink” instead of just “sink”. *Data_type*: bool, optional
`-update_timing`	When specified, the report includes detailed timing information including pin slew, capacitance, and location along with the clock tree structure. *Data_type*: bool, optional

Examples

- Use the following command to report the structure of the clock network for the specified clock trees, `m_clk` and `m_digit_clk`:

```
report_clock_tree_structure -clock_trees {m_clk m_digit_clk}
```

A part of the report is shown below.

```
Clock tree m_clk:
Total FF: 115
Max Level: 5
  (L1) TEST_CONTROL_INST/g137/ZN (ND2D1BWP)
  \_  (L1) port TEST_CONTROL_INST/m_clk
       \_ ... (21 sinks omitted)
       \_  (L2) DMA_INST/CPF_LS_158_m_clk/I -> Z (LVLHLD2BWP)
        |   \_ ... (8 sinks omitted)
        |   \_  (L3) DMA_INST/RC_CG_HIER_INST0/RC_CGIC_INST/CP -> Q (CKLNQD1BWP)
        |       \_ ... (8 sinks omitted)
        \_  (L2) RESULTS_CONV_INST/CPF_LS_159_m_clk/I -> Z (LVLHLD2BWP)
```

```
        |   \_ ... (15 sinks omitted)
        |   \_  (L3) RESULTS_CONV_INST/RC_CG_HIER_INST1/RC_CGIC_INST/CP -> Q (CKLNQD1BWP)
        |   |   \_ ... (9 sinks omitted)
        |   \_  (L3) RESULTS_CONV_INST/RC_CG_HIER_INST2/RC_CGIC_INST/CP -> Q (CKLNQD1BWP)
        |   |   \_ ... (16 sinks omitted)
```

.....

```
Clock tree m_digit_clk:
 Total FF: 9
  Max Level: 3
    (L1) TEST_CONTROL_INST/g141/ZN (IOA21D1BWP)
    \_  (L1) port TEST_CONTROL_INST/m_digit_clk
         \_ ... (9 sinks omitted)
```

- Use the following command to report the structure of the clock network inlcuding the timing information for the specified clock tree, `m_clk`:

```
report_clock_tree_structure -clock_trees {m_clk} -update_timing
```

```
Clock tree m_clk:
TEST_CONTROL_INST/g137/ZN root output at (273.210,301.000), lib_cell ND2D1BWP, level 1, slew 9.719ns, wire_cap 0.096pF, load_cap 0.452pF
\_ port TEST_CONTROL_INST/m_clk port output, preserved by {sdc_constraint}, level 1, slew 0.000ns, wire_cap 0.000pF, load_cap 0.000pF
   \_ ... (21 sinks omitted)
   \_ DMA_INST/CPF_LS_158_m_clk/I logic input at (169.210,300.720), lib_cell LVLHLD2BWP, level 2, slew 9.720ns
   | DMA_INST/CPF_LS_158_m_clk/Z logic output at (169.770,300.860), lib_cell LVLHLD2BWP, level 2, slew 0.431ns, wire_cap 0.008pF,
     load_cap 0.006pF
   | \_ ... (8 sinks omitted)
   | \_ DMA_INST/RC_CG_HIER_INST0/RC_CGIC_INST/CP cgate input at (115.450,305.760), lib_cell CKLNQD1BWP, level 3, slew 0.431ns
   |    DMA_INST/RC_CG_HIER_INST0/RC_CGIC_INST/Q cgate output at (116.290,305.900), lib_cell CKLNQD1BWP, level 3, slew 0.298ns,
        wire_cap 0.004pF, load_cap 0.005pF
   |    \_ ... (8 sinks omitted)
   \_ RESULTS_CONV_INST/CPF_LS_159_m_clk/I logic input at (199.170,320.880), lib_cell LVLHLD2BWP, level 2, slew 9.720ns
   | RESULTS_CONV_INST/CPF_LS_159_m_clk/Z logic output at (199.730,321.020), lib_cell LVLHLD2BWP, level 2, slew 0.714ns,
     wire_cap 0.021pF, load_cap 0.015pF
```

- Use the following command to report the structure of the clock network with generated clock trees expanded inline for the specified clock trees, `m_clk` and `m_digit_clk`:

```
report_clock_tree_structure -clock_trees {m_clk m_digit_clk} -expand_generated_clock_trees inline
```

```
Clock tree m_clk:
Total FF: 115
Max Level: 5
(L1) TEST_CONTROL_INST/g137/ZN (ND2D1BWP)
\_  (L1) port TEST_CONTROL_INST/m_clk
   \_ ... (21 sinks omitted)
   \_   (L2) DMA_INST/CPF_LS_158_m_clk/I -> Z (LVLHLD2BWP)
   |    \_ ... (8 sinks omitted)
   |    \_ (L3) DMA_INST/RC_CG_HIER_INST0/RC_CGIC_INST/CP -> Q (CKLNQD1BWP)
   |        \_ ... (8 sinks omitted)
   \_   (L2) RESULTS_CONV_INST/CPF_LS_159_m_clk/I -> Z (LVLHLD2BWP)
   |    \_ ... (15 sinks omitted)
   |    \_ (L3) RESULTS_CONV_INST/RC_CG_HIER_INST1/RC_CGIC_INST/CP -> Q (CKLNQD1BWP)
   |   |    \_ ... (9 sinks omitted)
   |    \_ (L3) RESULTS_CONV_INST/RC_CG_HIER_INST2/RC_CGIC_INST/CP -> Q (CKLNQD1BWP)
   |   | \_ ... (16 sinks omitted)
   |    \_ (L3) RESULTS_CONV_INST/RC_CG_HIER_INST3/RC_CGIC_INST/CP -> Q (CKLNQD1BWP)
   |   |    \_ ... (16 sinks omitted)
   |    \_ (L3) RESULTS_CONV_INST/RC_CG_HIER_INST6/RC_CGIC_INST/CP -> Q (CKLNQD1BWP)
   |   |    \_ ... (6 sinks omitted)
   |    \_ (L3) RESULTS_CONV_INST/RC_CG_HIER_INST7/RC_CGIC_INST/CP -> Q (CKLNQD1BWP)
   |        \_ ... (8 sinks omitted)
   \_   (L2) SPI_INST/RC_CG_HIER_INST17/RC_CGIC_INST/CP -> Q (CKLNQD1BWP)
         \_ ... (8 sinks omitted)

Clock tree m_digit_clk:
Total FF: 9
Max Level: 3
(L1) TEST_CONTROL_INST/g141/ZN (IOA21D1BWP)
\_   (L1) port TEST_CONTROL_INST/m_digit_clk
     \_ ... (9 sinks omitted)
```

Related Topics

- `create_clock_tree_spec`
- `create_clock_tree`
- The following section in Clock Tree Synthesis chapter of *Innovus Stylus Common UI User Guide*
  - Clock Tree Network Structure

report_clock_trees

```
report_clock_trees
[-help]
[-clock_trees {list_of_clock_trees}]
[-delay_corners {list_of_delay_corners} | -views {list_of_views}]
[-histograms]
[-list_special_pins]
[-num_capacitance_violating_pins number_of_pins]
[-num_fanout_violating_nets number_of_nets]
[-num_length_violating_nets number_of_nets]
[-num_resistance_violating_nets number_of_nets]
[-num_transition_time_violating_pins number_of_pins]
[-out_file file_name]
[-skip_timing_update]
[-summary]
[-tcl_list]
[-use_previous_timing]
[-early_only | -late_only]
```

This command reports a summary of all defined clock trees. In particular, this report provides a summary of numbers of clock gates at different depths in each clock tree. Use the `report_clock_trees` command to monitor the amount of clock gating used in the design and the buffering used in the clock tree. You can run this command any time after defining or extracting clock trees.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `report_clock_trees` parameter. For a detailed description of the command and all of its parameters, use the man command: `man report_clock_trees`.
`-clock_trees {list_of_clock_trees}`	
	Specifies that the report will be restricted to the specified clock trees. *Data_type*: list, optional
`-delay_corners {list_of_delay_corners}`	
	Specifies that the report will be restricted to the specified delay corners. **Note**: The `-delay_corners` and `-views` parameters are mutually exclusive. *Data_type*: (delay_corner)+, optional
`-early_only`	Specifies that only the early half of the delay corners will be reported. *Data_type*: bool, optional
`-histograms`	Specifies that histograms of key information will be displayed. *Data_type*: bool, optional
`-late_only`	Specifies that only the late half of the delay corners will be reported. *Data_type*: bool, optional
`-list_special_pins`	Specifies that the report will list individual pins that are stop, ignore, min, or other special type of pins. *Data_type*: bool, optional
`-num_capacitance_violating_pins number_of_pins`	
	Specifies the maximum number of capacitance violating pins reported in the "Max capacitance violation summary across all clock trees" table of the report. By default, the software reports the top 10 violations. *Data_type*: int, optional
`-num_fanout_violating_nets number_of_nets`	

	Specifies the maximum number of violating fanout nets reported in the "Max fanout violation summary across all clock trees" table of the report. By default, the software reports the top 10 violations. *Data_type*: int, optional
`-num_length_violating_nets` *number_of_nets*	
	Specifies the maximum number of `max_source_to_sink_net_length` violating nets reported in the "Max length violation summary across all clock trees" table of the report. By default, the software reports the top 10 violations. *Data_type*: int, optional
`-num_resistance_violating_nets` *number_of_nets*	
	Specifies the maximum number of `max_source_to_sink_net_resistance` violating nets reported in the "Max resistance violation summary across all clock trees" table of the report. By default, the software reports the top 10 violations. *Data_type*: int, optional
`-num_transition_time_violating_pins` *number_of_pins*	
	Specifies the maximum number of worst slew violating pins that are reported in the "Slew violation summary across all clock trees" table. By default, the software reports the top 10 violations. *Data_type*: int, optional
`-out_file` *file_name*	Specifies that a copy of the contents of the report will be saved to the specified file name. *Data_type*: string, optional
`-skip_timing_update`	Specifies that the timing engine should not be updated. The timing engine is used to calculate the timing data that is displayed in the report. By default, when you generate the report, the timing data is recalculated. This increases the time taken to generate the report. Specify this parameter to skip calculation of timing data. You can also specify the `-use_previous_timing` option to reuse the existing timing data for the report. *Data_type*: bool, optional
`-summary`	Specifies that just the summary table should be displayed in the report. *Data_type*: bool, optional
`-tcl_list`	Specifies that the report will be created as Tcl data instead of a text report. *Data_type*: bool, optional
`-use_previous_timing`	Specifies that the timing engine should not be invalidated. The timing engine is used to calculate the timing data that is displayed in the report. By default, this is invalidated when you generate the report, so all the timing data is recalculated. This increases the time taken to generate the report. Use this option to reuse the existing timing data for the report. *Data_type*: bool, optional
`-views {`*list_of_views*`}`	Restricts the report to the delay corners associated with the specified views. **Note**: The `-delay_corners` and `-views` parameters are mutually exclusive. *Data_type*: (analysis_view)+, optional

Examples

Example1: **Clock Timing Summary table**

```
Clock Timing Summary:
=====================

Target and measured clock slews (in ns):

-------------------------------------------------------------------------------------------------------------------------------------------------------
Clock tree   Timing Corner                      Worst Rising  Worst Falling  Worst Rising  Worst Falling  Leaf Slew      Leaf Slew  Trunk Slew     Trunk Slew
                                                Leaf Slew     Leaf Slew      Trunk Slew    Trunk Slew     Target Type    Target     Target Type    Target
-------------------------------------------------------------------------------------------------------------------------------------------------------
m_clk        AV_HL_FUNC_MAX_RC1_dc:setup.early     9.673         10.980         9.668         10.974     ignored        -          ignored        -
m_clk        AV_HL_FUNC_MAX_RC1_dc:setup.late      9.706         10.990         9.701         10.985     auto computed  -          auto computed  -
m_clk        AV_HL_FUNC_MIN_RC1_dc:hold.early      4.361          4.325         4.349          4.310     ignored        -          ignored        -
m_clk        AV_HL_FUNC_MIN_RC1_dc:hold.late       4.379          4.328         4.368          4.313     ignored        -          ignored        -
m_digit_clk  AV_HL_FUNC_MAX_RC1_dc:setup.early     0.214          0.244           -              -       ignored        -          ignored        -
m_digit_clk  AV_HL_FUNC_MAX_RC1_dc:setup.late      0.229          0.246           -              -       auto computed  -          auto computed
-------------------------------------------------------------------------------------------------------------------------------------------------------

* - indicates that target was not met.

auto extracted - target was extracted from SDC.
auto computed - target was computed when balancing trees.
```

Example2: Slew violation summary table across all clock trees for top 10 violations

```
Slew violation summary across all clock trees - Top 10 violating pins:
======================================================================

Target and measured clock slews (in ns):

-------------------------------------------------------------------------------------------------------------------------------------------
Half corner                      Violation  Slew    Slew      Dont   Ideal  Target           Pin
                                 amount     target  achieved  touch  net?   source
                                                              net?
-------------------------------------------------------------------------------------------------------------------------------------------
AV_HL_FUNC_MAX_RC1_dc:setup.late   10.171    0.814   10.985    N      N      liberty explicit  ARB_INST/present_state_reg[2]/CP
AV_HL_FUNC_MAX_RC1_dc:setup.late   10.171    0.814   10.985    N      N      liberty explicit  ARB_INST/present_state_reg[1]/CP
AV_HL_FUNC_MAX_RC1_dc:setup.late   10.171    0.814   10.985    N      N      liberty explicit  ARB_INST/dma_grant_reg/CP
AV_HL_FUNC_MAX_RC1_dc:setup.late   10.171    0.814   10.985    N      N      liberty explicit  ARB_INST/present_state_reg[0]/CP
AV_HL_FUNC_MAX_RC1_dc:setup.late   10.171    0.814   10.985    N      N      liberty explicit  ARB_INST/tdsp_grant_reg/CP
AV_HL_FUNC_MAX_RC1_dc:setup.late   10.171    0.814   10.985    N      N      liberty explicit  SPI_INST/present_state_reg[2]/CP
AV_HL_FUNC_MAX_RC1_dc:setup.late   10.171    0.814   10.985    N      N      liberty explicit  SPI_INST/present_state_reg[1]/CP
AV_HL_FUNC_MAX_RC1_dc:setup.late   10.171    0.814   10.985    N      N      liberty explicit  SPI_INST/dflag_reg/CP
AV_HL_FUNC_MAX_RC1_dc:setup.late   10.171    0.814   10.985    N      N      liberty explicit  SPI_INST/present_state_reg[0]/CP
AV_HL_FUNC_MAX_RC1_dc:setup.late   10.171    0.814   10.985    N      N      liberty explicit  SPI_INST/RC_CG_HIER_INST17/RC_CGIC_INST/CP
-------------------------------------------------------------------------------------------------------------------------------------------

Target sources:
auto extracted - target was extracted from SDC.
auto computed - target was computed when balancing trees.
explicit - target is explicitly set via cts_target_max_transition_time property.
pin explicit - target is explicitly set for this pin via cts_pin_target_max_transition_time property.
liberty explicit - target is explicitly set via max_transition from liberty library.

Found 0 pins on nets marked dont_touch that have slew violations.
Found 0 pins on nets marked dont_touch that do not have slew violations.
Found 0 pins on nets marked ideal_network that have slew violations.
Found 0 pins on nets marked ideal_network that do not have slew violations.
```

Example3: Max capacitance violation summary table for top 4 violations

```
Max capacitance violation summary across all clock trees - Top 2 violations:
============================================================================

Target and measured capacitances (in pF):

-----------------------------------------------------------------------------------------------------------------------
Half corner                      Violation Capacitance Capacitance Target                 Pin
                                 amount    target      achieved    source
-----------------------------------------------------------------------------------------------------------------------
AV_HL_FUNC_MAX_RC1_dc:setup.late 0.505     0.043       0.548   library_or_sdc_constraint TEST_CONTROL_INST/g137/ZN
AV_HL_FUNC_MAX_RC1_dc:setup.late 0.127     0.041       0.169   library_or_sdc_constraint TEST_CONTROL_INST/g142/ZN
-----------------------------------------------------------------------------------------------------------------------
Target sources:
target_max_capacitance_property - the target was set in the target_max_capacitance property
library_or_sdc_constraint - the non-frequency-dependent target was set in a library file or by an SDC constraint
frequency_dependent_library_or_sdc_constraint - the frequency-dependent target was set in a library file or by an SDC constraint
computed_from_slew_target - the target was calculated based on the slew target at a clock root
clock_root_forced - the target was forced to be zero at a clock root


Found a total of 0 clock tree nets with max resistance violations.
Found a total of 0 clock tree nets with max length violations.
Found a total of 0 clock tree nets with max fanout violations.
Found a total of 279 clock tree pins with a slew violation.
```

Example4: Count of violations table

```
Count of violations across all clock trees:
===========================================

-------------------------------------------------------------------------------
```

```
Clock Tree  #Max capacitance #Max resistance #Max length #Max fanout #Slew
Name        violations        violations     violations  violations  violations
--------------------------------------------------------------------------------
m_clk        1               0              0           0           30
m_digit_clk  0               0              0           0           0
m_dsram_clk  1               0              0           0           2
m_ram_clk    1               0              0           0           2
m_rcc_clk    0               0              0           0           69
m_spi_clk    0               0              0           0           0
refclk       0               0              0           0           0
refclk<1>    1               0              0           0           176
--------------------------------------------------------------------------------
Total        4               0              0           0           279
--------------------------------------------------------------------------------

Note the above table per clock tree is based on CCOpt clock tree view. The violations are counted across half corners.

Found a total of 4 clock tree pins with max capacitance violations.
Found a total of 4 max capacitance violations.
Total violation amount 0.838pF.
```

Example5: Clock Sink Summary Report

```
Clock Sink Summary:
===================

-------------------------------------------------------------------------------------------------------------------------------------------
Clock Tree   Explicit      Implicit      Explicit      Implicit      Explicit      Implicit      Posedge     Negedge     Memory     Enable     Non enable     Other
Name         exclude pins  exclude pins  ignore pins   ignore pins   stop pins     stop pins     Flops       Flops       Clock      Latch      Latch          Sinks
                                                                                                                       Pins       Sinks      Sinks
-------------------------------------------------------------------------------------------------------------------------------------------
m_clk              0             0             1             0             0             0           113           0           0          0              0           2
m_digit_clk        0             0             0             0             0             0             9           0           0          0              0           0
m_dsram_clk        0             0             0             0             0             0             0           0           1          0              0           0
m_ram_clk          0             0             0             0             0             0             0           0           1          0              0           0
m_rcc_clk          0             0             0             0             0             0           129           0           0          0              0           0
m_spi_clk          0             0             0             0             0             0            12           0           0          0              0           0
refclk             0             0             1             0             0             0           258           0           0          0              0           1
refclk<1>          0             0             1             0             0             0           258           0           0          0              0           1
-------------------------------------------------------------------------------------------------------------------------------------------

Clock Sink Summary across all clock trees:
==========================================

-----------------------------------------------------------------------------------------------------------------
Explicit      Implicit      Explicit      Implicit      Explicit      Implicit      Posedge     Negedge     Memory     Enable     Non enable     Other
exclude pins  exclude pins  ignore pins   ignore pins   stop pins     stop pins     Flops       Flops       Clock      Latch      Latch          Sinks
                                                                                                          Pins       Sinks      Sinks
-----------------------------------------------------------------------------------------------------------------
     0             0             3             0             0             0           521           0           2          0              0           3
-----------------------------------------------------------------------------------------------------------------
```

Related Information

- create_clock_tree_spec
- create_clock_tree
- The following section in Clock Tree Synthesis chapter of *Innovus Stylus Common UI User Guide*
  - Reporting

report_cts_cell_filtering_reasons

```
report_cts_cell_filtering_reasons
[-help]
[-cell_type buffer | inverter | logic | delay | clock_gate]
[-clock_trees {list_of_clock_trees}]
[-out_file file_name]
[-power_domain power_domain_name]
[-tcl_list]
```

Reports the reasons for filtering the specified cell types. The report can be saved as a file by specifying the `-out_file` parameter of the command or it can be viewed as a Tcl list by specifying the `-tcl_list` parameter. The output of the command is also included in the log.

In the report, "all" is used for the Clock tree names where the reason is common to all clock_trees. Some of the reasons for filtering that are reported and their brief descriptions are provided below:

- `Unbalanced rise/fall delays` : The difference between rise and fall delays for the cell is bigger than 20%; or any of rise or fall timing arcs is missing in the library.
- `Too tall`: The cell height is bigger than the maximum cell height specified using the `cts_max_instance_height` attribute.
- `Wrong power context` : The cell does not match the power context requirements. For example, do not have same power context on all pins.
- `Not available in all views` : The cell is not available in all analysis views because the library does not contain all the specified views (corner cases) in the design for that specific cell. To avoid this filtering, check the libraries and the MMMC configuration to see why the cell is not available in some views.
- `Always on but not in cpf`: The cell is marked as always on but does not match the CPF description. You can avoid this filtering by checking/updating the CPF.
- `Weak max cap cell`: The cell is unable to drive enough capacitance at certain clock frequencies.
- `Library trimming`: The cell was removed automatically because it is an inefficient cell. The cell was removed as an inefficient cell to reduce the library size and save runtime. To avoid removing such cells, disable library trimming.
- `Can't use due to library mismatch`: Check the library data for the cell - both timing and physical data for the cell are needed. There is a mismatch between LEF and liberty libraries because the cell is missing in one of them.
- `Invalid for balancing`: The cell is filtered out because it does not match several predefined properties for balancing, such as: 1) cell delay is too small; 2)cell has an invalid (rise/fall) timing arc.
- `No test enable signal`: The clock gate cell is filtered out because it does not have a test enable input.
- `Has test enable signal`: The clock gate cell is filtered out because it has a test enable input.

  Note: The above two cell filtering reasons are context dependent. The software filters all clock gates, reporting the clock gates with test enables as one set and the clock gates without test enables as another set. A clock gate either has a test enable or it does not have one, so all clock gates are listed in one of the two sets. However, if you need a clock gate with no test enable signal, but there are no such clock gates, you can use a clock gate that has a test enable signal. But the other way around is not possible, which means if you need a clock gate with test enable signal but there is no such clock gate, you cannot use one that does not have a test enable signal.
- `Has observable output`: Clock gates without observability outputs are preferred, if they are available. Normally any cells with observability outputs will be filtered out. But if all the available cells have observability outputs they will not be filtered.
- `Edge sensitive`: The clock gate cell is filtered out because it is not positive edge sensitive.
- `Cannot be legalized`: The cell is filtered because it cannot physically be legalized in the power domain.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `report_cts_cell_filtering_reasons` parameter. For a detailed description of the command and all of its parameters, use the man command: man `report_cts_cell_filtering_reasons`.
`-cell_type buffer | inverter | logic | delay | clock_gate`	
	Restrict the report the specified library cell type. *Data_type*: enum, optional
`-clock_trees {`*`list_of_clock_trees`*`}`	
	Restricts the report to the specified clock trees. *Data_type*: string, optional
`-out_file` *`file_name`*	Specifies that a copy of the contents of the report will be saved to the specified file name. *Data_type*: string, optional
`-power_domain` *`power_domain_name`*	Restricts the report to the specified power domain. *Data_type*: (power_domain), optional
`-tcl_list`	Returns the report as Tcl formatted data, and formats the `-file` as tcl, if specified. *Data_type*: bool, optional

Examples

- Use the following command to filter reasons for clock gate cell types:

```
report_cts_cell_filtering_reasons -cell_type clock_gate

Filtering reasons for cell type: clock gate cell
================================================

-------------------------------------------------------------------------------------------------
Clock trees  Power domain   Reason                  Library cells
-------------------------------------------------------------------------------------------------
{ clk4 }     auto-default   No test enable signal  { TLATNCAX12 TLATNCAX16 TLATNCAX2 TLATNCAX20
                                                      TLATNCAX3 TLATNCAX4 TLATNCAX6
                                                      TLATNCAX8 }
{ clk4 }     auto-default   Has test enable signal { TLATNTSCAX12 TLATNTSCAX16 TLATNTSCAX2
                                                      TLATNTSCAX20 TLATNTSCAX3 TLATNTSCAX4
                                                      TLATNTSCAX6 TLATNTSCAX8 }
-------------------------------------------------------------------------------------------------
```

- Use the following command to filter reasons for buffer cell types:

```
report_cts_cell_filtering_reasons -cell_type buffer

Example output format:

Filtering reasons for cell type: buffer
=======================================

-------------------------------------------------------------------------------
Clock trees   Power domain   Reason             Library cells
-------------------------------------------------------------------------------
all           AO             Library trimming  { BUFFD12BWP BUFFD2BWP BUFFD3BWP
                                                   BUFFD4BWP BUFFD6BWP BUFFD8BWP
                                                   PTBUFFD2BWP }
all           PLL            Unbalanced        { BUFFD12BWP BUFFD2BWP BUFFD3BWP
                             rise/fall delays    BUFFD4BWP BUFFD6BWP BUFFD8BWP
                                                 PTBUFFD2BWP }
-------------------------------------------------------------------------------
```

- Use the following command to filter reasons for inverter cell types:

```
report_cts_cell_filtering_reasons -cell_type inverter

Example output format:

Filtering reasons for cell type: inverter
```

```
====================================

-------------------------------------------------------------------------------
Clock trees   Power domain   Reason              Library cells
-------------------------------------------------------------------------------
all           AO             Unbalanced          { INVD12BWP INVD16BWP INVD1BWP
                             rise/fall delays      INVD2BWP INVD3BWP INVD4BWP
                                                   INVD6BWP INVD8BWP }
all           PLL           Library Trimming     { INVD12BWP INVD16BWP INVD1BWP
                                                   INVD2BWP INVD3BWP INVD4BWP
                                                   INVD6BWP}
-------------------------------------------------------------------------------
```

Related Information

- Clock Tree Synthesis chapter in the *Innovus Stylus Common UI User Guide*
  - Reporting

report_pin_insertion_delays

```
  report_pin_insertion_delays
[-help]
[-bin_size string]
[-check_type {setup | hold}]
[-clock_trees {string1 string2 ...}]
[-delay_corner delay_corner_name]
[-delay_type {early | late}]
[-out_file file_name]
[-skew_groups {string1 string2 ...} | -include_reporting_only_skew_groups]
[-sources {string1 string2 ...}]
```

Reports pin insertion delays at clock tree sinks. By default, the report visits all the clock tree sinks, and checks for pin insertion_delay at each sink. Statistics about the distribution of pin insertion_delay values are generated and reported. The intention of the report is to give a quick overview of the distribution and nature of the pin insertion delays found, which can be used to identify the problem areas such as excessively large skews, surprisingly numerous skews, or large numbers of small skews, and so on.

The `-clock_trees` and `-skew_groups` parameters are used to restrict the scope of the report as a subset of the clock tree sinks, which means that when the `-clock_trees` parameter is specified, only those sinks that are under the listed clock trees are selected, and when the `-skew_groups` parameter is specified, only those sinks that are under the listed skew groups are selected. By default, the report analyzes the pin insertion delays that are found under the CCOpt primary half corner. The `-delay_corner`, `-check_type`, and `-delay_type` parameters permit the selection of another half corner to report against.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `report_pin_insertion_delays` parameter. For a detailed description of the command and all of its parameters, use the man command: `man report_pin_insertion_delays`.
`-bin_size string`	Specifies a bin (bucket) size for the report histograms. By default the software uses the histogram bin size specified using the cts category attribute, `cts_pin_insertion_delay_histogram_bin_size`. Use this parameter to override this default bin size value. *Data_type*: string, optional
`-check_type {setup | hold}`	
	Specifies the check type for the specified skew group to be reported. You can specify either `setup` or `hold`. If a check type is specified, but the `-delay_type` parameter is not specified, the software behaves in the following manner: • `-check_type -setup` will mean `-delay_type -late` • `-check_type -hold` will mean `-delay_type -early` **Note** : If a check type is not specified, the software will always look at what the views permit, preferring the “setup” view if there is a choice. *Data_type*: enum, optional
`-clock_trees {string1 string2 ...}`	
	Restricts the report to the specified clock trees. This means that only those sinks that are under the specified clock trees are selected for reporting. *Default*: all defined clock trees *Data_type*: list, optional
`-delay_corner delay_corner_name`	
	Reports in the specified delay corner. By default, the software will use the primary-half delay corner that is set using the CCOpt attribute, `cts_primary_delay_corner` . *Data_type*: (delay_corner), optional
`-delay_type {early | late}`	

	Reports in the specified delay type, early or late. When late is specified, only the late half of the delay corners are reported. When early is specified, only the early half of the delay corners are reported. *Default:* `late` *Data_type*: enum, optional
`-include_reporting_only_skew_groups`	When specified, the report additionally includes the reporting-only skew groups. A reporting-only skew group is a skew group whose `cts_skew_group_constrains` attribute is set to `none`. Such a skew group imposes no clock balancing constraint and will not be considered by CTS. By default, the reporting-only skew groups are not included in the report. If a reporting-only skew group is explicitly specified using the `-skew_groups` parameter, then that skew group is always included in the report output. **Note**: The `-include_reporting_only_skew_groups` and `-skew_groups` parameters are mutually exclusive. If both these parameters are specified together, the software errors out. *Data_type*: bool, optional
`-out_file` *file_name*	Specifies that a copy of the contents of the report be saved to the specified file name. *Data_type*: string, optional
`-skew_groups {string1 string2 ...}`	
	Restricts the report to the specified skew groups. This means that only those sinks that are under the specified skew groups are selected for reporting. **Note**: The `-include_reporting_only_skew_groups` and `-skew_groups` parameters are mutually exclusive. If both these parameters are specified together, the software errors out. *Default*: all defined skew groups *Data_type*: list, optional

`-sources {string1 string2 ...}`	Restricts the report to skew group sink pins that have a pin insertion delay value from one of the listed sources.
	This `-sources` list of categories takes wildcard values that are expanded to matching categories.
	For example:
	`report_pin_insertion_delays -sources u*`
	This is equivalent to:
	`report_pin_insertion_delays -sources {useful_skew user}`
	Data_type: list, optional

Example

The following command reports the pin insertion delays at all clock tree sinks.

```
report_pin_insertion_delays
```

Note: The CCOpt pin insertion delays are +ve valued if they produce an advance, and -ve valued if they produce a delay. This can be viewed in the report output shown below.

```
Positive (advancing) pin insertion delays
=========================================

Found 1 advances (2.041% of 49 clock tree sinks)

-------------------------------------------------
From (ns)     To (ns)     Count
-------------------------------------------------
0.000         0.010       1
-------------------------------------------------

Mean              : 0.006ns
Std.Dev           : 0.000ns

Smallest advance  : 0.006ns at d2211a/CK
```

```
Largest advance   : 0.006ns at d2211a/CK

Negative (delaying) pin insertion delays
================================================

Found 23 delays (46.939% of 49 clock tree sinks)

------------------------------------------------
From (ns)     To (ns)    Count
------------------------------------------------
-0.120       -0.110       1
-0.110       -0.100       0
-0.100       -0.090       1
-0.090       -0.080       2
-0.080       -0.070       3
-0.070       -0.060       2
-0.060       -0.050       4
-0.050       -0.040       4
-0.040       -0.030       3
-0.030       -0.020       1
-0.020       -0.010       2
------------------------------------------------

Mean              : -0.056ns
Std.Dev           : 0.025ns

Smallest delay    : -0.014ns at d3211a/CK
Largest delay     : -0.110ns at d1111d/CK
```

Related Information

- Clock Tree Synthesis chapter in the *Innovus Stylus Common UI User Guide*
  - Reporting

report_skew_groups

```
report_skew_groups
[-help]
[-delay_corners {list_of_delay_corners} | -views {list_of_views}]
[-exclude_pin_insertion_delay]
[-exclude_source_latency]
[-out_file file_name]
[-from sources]
[-histograms]
[-num_paths number_of_paths]
[-skew_groups list_of_skew_groups | -include_reporting_only_skew_groups]
[-summarize_stage_depth cell_type_list]
[-skip_timing_update]
[-summary]
[-through list_of_pins]
[-tcl_list]
[-to list_of_sinks]
[-use_previous_timing]
[-early_only | -late_only ]
```

This command displays information about skew and insertion delay in skew groups. By default, the `report_skew_groups` command displays two summary tables at the beginning of the report. The first table lists the insertion delays and skews (along with target values) for each skew group in each timing corner. The second table lists the start and end points of the min and max paths in the skew group. The remainder of the report contains detailed min and max path information for each skew group, in each timing corner.

You can customize the columns included in the skew group report by setting the attribute, `cts_skew_group_report_columns`.

You can customize the size of the histogram ranges by setting the attribute, `cts_skew_group_report_histogram_bin_size`.

Note: For more information about these attributes, see the cts Category Attributes chapter.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `report_skew_groups` parameter. For a detailed description of the command and all of its parameters, use the man command: man `report_skew_groups`.
`-delay_corners list_of_delay_corners`	
	Specifies that the report will be restricted to the specified delay corners. **Note**: The `-delay_corners` and `-views` parameters are mutually exclusive. *Data_type*: (delay_corner)+, optional
`-early_only`	Specifies that only the early half of the delay corners will be reported. *Data_type*: bool, optional
`-exclude_pin_insertion_delay`	
	Specifies that pin insertion delay should be excluded from the report. *Data_type*: bool, optional
`-exclude_source_latency`	Specifies that source latency of the clocks should be excluded from the report. *Data_type*: bool, optional
`-from sources`	Specifies the list of sources of the skew group to consider for the report. If specified, only paths beginning at one of the sources specified will be reported. *Data_type*: string, optional
`-histograms`	Specifies that histograms of key information will be displayed. *Data_type*: bool, optional

`-include_reporting_only_skew_groups`	When specified, the report additionally includes the reporting-only skew groups. A reporting-only skew group is a skew group whose `cts_skew_group_constrains` attribute is set to `none`. Such a skew group imposes no clock balancing constraint and will not be considered by CTS. By default, the reporting-only skew groups are not included in the report. If a reporting-only skew group is explicitly specified using the `-skew_groups` parameter, then that skew group is always included in the report output. **Note**: The `-include_reporting_only_skew_groups` and `-skew_groups` parameters are mutually exclusive. If both these parameters are specified together, the software errors out. *Data_type*: bool, optional
`-late_only`	Specifies that only the late half of the delay corners will be reported. *Data_type*: bool, optional
`-num_paths` *number_of_paths*	Specifies the number of paths to report on for each timing corner. *Default*: 1 *Data_type*: int, optional
`-out_file` *file_name*	Specifies that a copy of the contents of the report will be saved to the specified file name. *Data_type*: string, optional
`-skew_groups` *list_of_skew_groups*	
	Restricts the report to the specified skew groups. By default, all defined skew groups are displayed. *Data_type*: string, optional
`-skip_timing_update`	Specifies that the timing engine should not be updated. The timing engine is used to calculate the timing data that is displayed in the report. By default, when you generate the report, all the timing data is recalculated. This increases the time taken to generate the report. Use this option to skip calculating the timing data. You can also use the `-use_previous_timing` parameter to reuse the existing timing data for the report. *Data_type*: bool, optional
`-summarize_stage_depth` *cell_type_list*	

	Specifies that the command will consider the number of cells of the specified type(s) in the tree, instead of considering delay while reporting. Possible values are buffer, inverter,logic, clock_gate, and generator. You can also specify * to include all cell types.
	The following command shows a report of the skew groups with counts of the buffers and inverters, instead of the delay.
	`report_skew_groups -summarize_stage_depth {buffer inverter}`
	Note: The * setting is not used in combination with any other cell type. For example, the below is not a valid setting.
	`-summarize_stage_depth {* buffer}`
	Data_type: list, optional

`-summary`	Specifies that just the summary table should be displayed in the report. *Data_type*: bool, optional
`-tcl_list`	Specifies that the report will be created as Tcl data instead of a text report. *Data_type*: bool, optional
`-through` *list_of_pins*	Provides a list of pin objects representing points that all # paths reported must pass through. The paths must pass through one of the pins specified. *Data_type*: string, optional
`-to` *list_of_sinks*	Provides a list of pin objects representing all the sinks of skew groups to consider for the report. If specified, only paths terminating at one of the sinks specified will be reported. *Data_type*: string, optional

`-use_previous_timing`	Specifies that the timing engine should not be invalidated. The timing engine is used to calculate the timing data that is displayed in the report. By default, this is invalidated when you generate the report, so all the timing data is recalculated. This increases the time taken to generate the report. Use this option to reuse the existing timing data for the report. *Data_type*: bool, optional
`-views {`*`list_of_views`*`}`	Restricts the report to the delay corners associated with the specified views. **Note**: The `-delay_corners` and `-views` parameters are mutually exclusive. *Data_type*: (analysis_view)+, optional

Examples

Use the following command to display the skew group information as a summary table in the report:

Example1: Use the following command to display the skew group information as a summary table in the report:

```
report_skew_groups -summary
Skew Group Structure:
=====================

-------------------------------------------------------------------------
Skew Group                Sources    Constrained Sinks    Unconstrained Sinks
-------------------------------------------------------------------------
m_clk/PM_HL_FUNC              1              114                    1
m_digit_clk/PM_HL_FUNC        1                9                    0
m_dsram_clk/PM_HL_FUNC        1                1                    0
m_ram_clk/PM_HL_FUNC          1                1                    0
m_roc_clk/PM_HL_FUNC          1              129                    0
m_spi_clk/PM_HL_FUNC          1               12                    0
refclk/PM_HL_FUNC             1              258                    1
-------------------------------------------------------------------------
Skew Group Summary:
===================

-------------------------------------------------------------------------------------------------------------------------------------------------
Timing Corner                        Skew Group              ID Target  Min ID  Max ID  Avg ID  Std.Dev.ID  Skew Target Type  Skew Target  Skew  Skew window occupancy
-------------------------------------------------------------------------------------------------------------------------------------------------
AV_HL_FUNC_MAX_RC1_dc:setup.early    m_clk/PM_HL_FUNC            -     0.039   3.924   2.851   1.390       ignored           -            3.884 -
                                     m_digit_clk/PM_HL_FUNC      -     0.000   0.001   0.001   0.000       ignored           -            0.000 -
                                     m_dsram_clk/PM_HL_FUNC      -     1.930   1.930   1.930   0.000       ignored           -            0.000 -
                                     m_ram_clk/PM_HL_FUNC        -     0.031   0.031   0.031   0.000       ignored           -            0.000 -
                                     m_roc_clk/PM_HL_FUNC        -     0.650   1.377   1.236   0.105       ignored           -            0.727 -
                                     m_spi_clk/PM_HL_FUNC        -     0.301   0.303   0.302   0.001       ignored           -            0.003 -
                                     refclk/PM_HL_FUNC           -     1.446   2.683   2.311   0.417       ignored           -            1.237 -
AV_HL_FUNC_MAX_RC1_dc:setup.late     m_clk/PM_HL_FUNC            none  0.039   3.932   2.857   1.393       auto computed     -            3.893 -
                                     m_digit_clk/PM_HL_FUNC      none  0.001   0.001   0.001   0.000       auto computed     -            0.000 -
-------------------------------------------------------------------------------------------------------------------------------------------------
* - indicates that target was not met.

Skew Group Min/Max path pins:
=============================

-------------------------------------------------------------------------------------------
Timing Corner                        Skew Group              Min ID    PathID    Max ID    PathID
-------------------------------------------------------------------------------------------
AV_HL_FUNC_MAX_RC1_dc:setup.early    m_clk/PM_HL_FUNC        0.039     -         3.924     -
-    min ARB_INST/dma_grant_reg/CP
-    max RESULTS_CONV_INST/high_mag_reg[13]/CP
                                     m_digit_clk/PM_HL_FUNC  0.000     -         0.001     -
-    min DIGIT_REG_INST/digit_out_reg[0]/CP
-    max DIGIT_REG_INST/digit_out_reg[6]/CP
                                     m_dsram_clk/PM_HL_FUNC  1.930     -         1.930     -
-    min RAM_256x16_TEST_INST/RAM_256x16_INST/CLK
-    max RAM_256x16_TEST_INST/RAM_256x16_INST/CLK
                                     m_ram_clk/PM_HL_FUNC    0.031     -         0.031     -
-    min RAM_128x16_TEST_INST/RAM_128x16_INST/CLK
-    max RAM_128x16_TEST_INST/RAM_128x16_INST/CLK
                                     m_roc_clk/PM_HL_FUNC    0.650     -         1.377     -
-    min RESULTS_CONV_INST/go_reg/CP
-    max RESULTS_CONV_INST/r941_reg[15]/CP
                                     m_spi_clk/PM_HL_FUNC    0.301     -         0.303     -
```

Note:

- The “Timing Corner” column specifies the delay corner, timing check type (setup or hold) and derate type (early or late) reporting conditions for the skew group. All *early corners are reported using early clock derates and all *late corners are reported using late clock derates. A skew group row with no timing column specification is using the timing corner specified above it.
- The "Skew Target Type" column shows either "explicit", “auto computed” or “ignored”. Only the timing corner(s) with "explicit" or "auto computed" are actually considered during clock tree construction, and only these corners will have a numeric skew target in the “Target Skew” column. Generally, only timing corners with late derates for a delay corner are activated for clock tree construction.
- The "Skew window occupancy" column is defined as the percentage of sinks in the skew group that fall within the user-defined skew target. It is computed by finding a min and max latency window in which the largest proportion of sinks fall. The intent of this column is to indicate, when the skew target is not met, some measure of how badly it has not been met by showing the largest proportion of sinks that occupy the desired skew target.

Example2: Use the following command to shows a report of the skew groups with counts of the buffers and inverters, instead of the delay:

```
report_skew_groups -summarize_stage_depth {buffer inverter} -skew_groups m_dsram_clk/PM_HL_FUNC

Timing for timing corner AV_HL_FUNC_MIN_RC1_dc:hold.late, max clock_path:
=========================================================================

 PathID : 8
```

```
Path type : skew group m_dsram_clk/PM_HL_FUNC (path 1 of 1)
Start : TEST_INST/g139/ZN
End : RAM_TEST_INST/RAM_INST/CLK
Counted cell types : buffer inverter
Count : 0

-------------------------------------------------------------------------------------------------------------------
Name                        Lib cell  Event Incr Count Slew Cap    Location Distance Fanout Status Clockpath trace
                                                      (ns) (pF)     (um)
-------------------------------------------------------------------------------------------------------------------
TEST_INST/g139/ZN         - IOA21D1BWP rise  -    0   0.131 0.015 (139.930,130.200)   -       1
RAM_TEST_INST/LS_m_clk/I  - LVLHLD2BWP rise  0    0   0.131 -     (250.130,127.960)  112.440  -
-------------------------------------------------------------------------------------------------------------------
```

Related Information

- create_skew_group
- delete_skew_groups
- get_db
- get_skew_group_path
- get_skew_group_delay
- set_db
- update_skew_group
- The following sections in Clock Tree Synthesis chapter of *Innovus Stylus Common UI User Guide*
  - Clock Trees and Skew Groups
  - Manual Setup and Adjustment of the Clock Specification
  - Reporting

reset_cts_config

```
reset_cts_config
[-help]
[-preserve_sink_insertion_delays]
```

Resets the CTS configuration. This command wipes out all CTS-specific data in Innovus. However, you can preserve the pin insertion delays on clock tree sinks by specifying the `-preserve_sink_insertion_delays` parameter.

Parameter

`-help`	Outputs a brief description of the `reset_cts_config` parameter.
`-preserve_sink_insertion_delays`	When specified, preserves pin insertion delays currently configured on clock tree sinks. *Data_type*: bool, optional

Example

- The following command resets the CTS configuration:
  ```
  reset_cts_config
  ```

Related Commands

- `check_cts_config`
- `read_cts_config`
- `write_cts_config`
- `create_clock_tree_spec`
- `delete_clock_tree_spec`
- `delete_clock_trees`
- `delete_skew_groups`

reset_clock_tree_routing_state

```
reset_clock_tree_routing_state
[-help]
[-excluded {netname1 netname2.....}]
[-no_delete_routes]
[-reset_preroutes]
```

Resets the status of the clock nets to "routed". The default behavior of the software is to then delete these nets, but you can specify the `-no_delete_routes` parameter to stop these nets from being deleted. The purpose of this command is to reset the routing state post NanoRoute, so that it can be rerun later using the `route_clock_tree_nets` command.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `reset_clock_tree_routing_state` parameter. For a detailed description of the command and all of its parameters, use the man command: `man reset_clock_tree_routing_state`.
`-excluded {netname1 netname2.....}`	Specifies the list of nets whose routing state is not to be reset. *Data_type*: (net)+, optional
`-no_delete_routes`	
	Specifies that the nets whose status is set to routed should not be deleted. *Data_type*: bool, optional
`-reset_preroutes`	
	When specified, resets the status of pre routes. By default, the pre-routes will be skipped. *Data_type*: bool, optional

Example

- The following command resets the routing state of clock nets without deleting them and excluding the specified net:

  ```
  reset_clock_tree_routing_state -no_delete_routes -excluded {clock1111}
  ```

Related Command

- `route_clock_tree_nets`

route_flexible_htrees

```
route_flexible_htrees
[-help]
[nets net+]
```

Routes CCOpt flexible h-tree nets as a separate step after running the synthesize_flexible_htrees command. The following use models are supported:

1. Routing during H-tree syntheses:
    a. `create_flexible_htree`
    b. `synthesize_flexible_htrees`
2. Routing after H-tree synthesis (block level)
    a. `create_flexible_htree`
    b. `synthesize_flexible_htrees -use_estimated_routes`
    c. `route_flexible_htrees`

Note: When using this command, the software expects the input routes of nets in the flexible H-tree to be fully connected, as it is the case for routes generated by `synthesize_flexible_htrees -use_estimated_routes`. If this expectation is not met, then unexpected results such as open nets may occur and you should check for connectivity-related warnings in the log file.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `route_flexible_htrees` parameter. For a detailed description of the command and all of its parameters, use the man command: `man route_flexible_htrees`.
`-nets net+`	
	Specifies a list of flexible h-tree nets to route. The specified nets must all have been created by the `synthesize_flexible_htrees` command. By default, the software automatically determines all the flexible h-tree nets that need to be routed.

Related Commands

- `create_flexible_htree`
- `synthesize_flexible_htrees`

Related Attributes

You can also set the following attributes on the flexible H-tree design objects by using the `set_db attribute_name value` command:

- `cts_flexible_htree_placement_legalization_effort`
- `flexible_htrees`

For detailed information, see the cts Category Attributes chapter in the *Innovus Stylus Common UI User Guide*. You can also use the following command:

```
get_db attribute_name
```

route_clock_tree_nets

```
route_clock_tree_nets
[-help]
[-mesh]
```

Performs routing of clock tree nets using the CTS route guides. Use this command after running `ccopt_design`.

CTS uses internal clock routes during the early stages of CTS. It then generates route guides that are passed to NanoRoute for final clock tree routing. This is done by default for full CTS runs. If, for some reason, you have disabled final clock tree routing in CTS, then this command can be used to complete the clock tree routing based on the CTS route guides. When the `-mesh` parameter is specified, the command only routes the mesh nets.

Note: Before running this command, ensure that the CTS data structures are in place.

Parameters

`-help`	Outputs a brief description for `route_clock_tree_nets` parameter.
`-mesh`	Specifies that only mesh nets should be routed. *Data_type*: bool, optional

Related Commands

- `ccopt_design`
- `reset_clock_tree_routing_state`

synthesize_flexible_htrees

```
synthesize_flexible_htrees
[-help]
[-block_master_clone]
[-dry_run]
[-partition_nets_dir dirname]
[-spec_file filename]
[-use_estimated_routes | -trial_mode {true | false | placement_only | routing_only}]
```

This command is used to synthesize all flexible H-trees that have been created but not previously synthesized, in one go. The flexible H-trees are created using the `create_flexible_htree` command. The `synthesize_flexible_htrees` command helps reduce the runtime by synthesizing all flexible H-trees in one go as compared to if the flexible H-trees were synthesized one at a time. This is because when you use this command, nanoRoute is called only once to route all flexible H-trees at the same time.

Note: You must run the `synthesize_flexible_htrees` command to synthesize any flexible H-trees previously created using the `create_flexible_htree` command.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `synthesize_flexible_htrees` parameter. For a detailed description of the command and all of its parameters, use the man command: `man synthesize_flexible_htrees`.
`-block_master_clone`	When specified, blocks master/clone partitions for H-tree instances and nets. If this parameter is not specified, and the master/clone partitions do not contain H-tree sink pins, then the software will error out with the following error: `**ERROR: (IMPCCOPT-1506): Flexible H-tree 'htree' uses a sink grid, which is not supported on master/clone designs unless master/clone partitions are blocked during flexible H-tree synthesis. If applicable, master/clone partitions can be blocked using synthesize_ccopt_flexible_htrees -block_master_clone.` *Data_type*: bool, optional
`-dry_run`	Runs the synthesis algorithm but do not insert the H-tree into the database. *Data_type*: bool, optional
`-partition_nets_dir` *dirname*	Specifies the name of the directory to which route guides and the files specifying the names of synthesized H-tree nets per partition/top module will be written. This parameter is used in master clone designs where H-tree routing needs to be done at the partition-level because of a lack of partition-awareness when running NanoRoute at the top level. For non-master clone designs, even if the design contains partitions, this data is generated only for the top module. *Data_type*: string, optional
`-spec_file` *fileName*	Specifies the name of the clock tree specification script file. When the file name is specified, the clock tree specification file is not amended to reflect the new state of the clock tree. Instead, the amendments, which are in the form of Tcl commands, are written to the file specified using this parameter. *Data_type*: string, optional
`-trial_mode {true | false | placement_only | routing_only}`	When set, enables trial routing and placement mode for flexible H-tree synthesis. The following settings are available: `true`: Run routing and detailed placement in trial mode. `false`: Run neither routing nor detailed placement in trial mode. THis si the default setting. `placement_only`: Only run detailed placement in trial mode. `routing_only` : Only run routing in trial mode. **Note**: The `-trial_mode` and `-use_estimated_routes` parameters are mutually exclusive. The `-trial_mode true` setting and `-use_estimated_routes` share the same behavior except that when `-trial_mode true` is specified, the software will skip full routing and treat estimated routes as fully-routed wires. When `-trial_mode` is set to `true`, `get_db flexible_htrees .nets.skip_routing` would be `true` When `-use_estimated_routes` is specified, `get_db flexible_htrees .nets.skip_routing` would be `false` *Default*: `false`
`-use_estimated_routes`	Specifies that NanoRoute should not be called. So, the routing estimates are left in the netlist. *Data_type*: bool, optional

Example

- The following set of commands are used to a flexible H-tree of the type, `INVX9` and will be placed in the specified sink grid. Other buffers added to the H-tree will be of type `INVX13`. The flexible H-tree is then synthesized using the `synthesize_flexible_htrees` command.

```
create_flexible_htree -name htreeC -sink_grid {6 6} -final_cell INVX9 -trunk_cell INVX13
synthesize_flexible_htrees
```

Related Information

- `ccopt_design`
- `create_flexible_htree`
- `delete_flexible_htrees`
- `get_db`
- `set_db`
- The following section in Clock Tree Synthesis chapter in the *Innovus Stylus Common UI User Guide*
  - Flexible H-Tree and Multi-Tap Clock Flow
- For details of the use model for this feature, see the Flexible H-tree and Multi-Tap Clock Flow in Innovus Application Note on the Cadence Online Support website.

update_clock_tree_source_latency

```
update_clock_tree_source_latency
[-help]
-clock_tree_source_groups source_group_list
```

Stores clock arrival and transition times at the clock tree source group roots in preparation for multi-tap CTS.

This command may be helpful for configuring arrival times for clock structures in which a mesh drives the source group roots. With other structures such as H-tree, which have a single driver per net, the source group roots are usually defined as generated clock trees and CTS will automatically propagate the clock arrival times from the clock source, through the drive structure, up to the inputs of the source group roots. Flexible H-tree synthesis takes care of all this automatically. However for mesh applications, the CTS clock specification, clock trees defined by the `create_clock_tree` and `create_generated_clock_tree` commands, cannot include nets with multiple drivers, so CTS cannot automatically propagate arrival times through the clock mesh. For these structures, the source group is typically defined as a collection of non-generated clock trees, and the arrival times are explicitly annotated at their roots before the start of CTS.

Note: The `update_clock_tree_source_latency` command queries the timing graph to get arrival times and transitions. If timing is not already available, the command will first perform a timing graph update, which may take some time. The command assumes that for a particular source group there is a single common clock phase present at all roots. If there is no common clock phase at all roots, or if there are multiple common clocks phases, the command will issue an error message and quit. Also, if a specified source group includes any generated clock trees, the command will issue an error message and quit.

Parameters

`-help`	Outputs a brief description of the `update_clock_tree_source_latency` command. For a detailed description of the command, use the man command: `man update_clock_tree_source_latency`.
`-clock_tree_source_groups source_group_list`	
	The list of clock tree source groups for whose clock roots the latencies will be stored. This parameter is required to be specified. *Data_type*: list, required

Example

The following command updates the latencies for a single source group, `four_taps`. In this case, timing is already available and no timing graph update is required.

```
update_clock_tree_source_latency -clock_tree_source_groups four_taps

Updating clock tree source latencies for source group 'four_taps'.

Found common clock 'clock' at all source group roots.

Set source timing for 4 roots (latency rising = [0.105ns, 0.116ns], falling = [1.130ns, 1.140ns]; transition = 0.004ns)
```

Related Information

- `create_clock_tree`
- `create_generated_clock_tree`

update_clock_tree_spec_annotations

```
update_clock_tree_spec_annotations
[-help]
[-out_file file_name]
```

Updates all ideal nets, transition and delay annotations to match active timing constraints and removes all prior annotations from the clock tree specification. When this command is specified, all prior annotations are removed from the clock tree specification and the `set_annotated_transition` and `set_annotated_delay` values are mapped to the `cts_annotated_*` attribute settings. The updated settings can be written to a file using the `-out_file` option or they can be applied immediately.

The following object attributes are provided to model the data captured by `set_ideal_network`, `set_annotated_delay`, `set_annotated_transition`, `set_ideal_latency` and `set_ideal_transition`:

- `cts_ideal_net`: If this attribute is set, the clock tree timing engine will consider the specified net as ideal.
  Default: `false`
- `cts_annotated_delay(pin)`: Overrides any clock tree timing engine computed cell arc or net arc delays to this pin, in a similar manner to SDC set_annotated_delay.
  Default: `{}`
- `cts_annotated_delay(port)`: Overrides any clock tree timing engine computed cell arc or net arc delays to this pin, in a similar manner to SDC set_annotated_delay.
  Default: `{}`
  Note:
  - **For cell arcs**: The format is dict where the keys are pins and the values are a rise fall delay pair. For example:
    `"<from pin 1> {<from rise delay 1> <from fall delay 1>} <from pin 2> {<from rise delay 1> <from fall delay 2>}"`
  - **For net arcs**: The format is dict where the keys are pins and the values are the delay. For example:
    `"<from pin 1> <delay 1> <from pin 2> <delay 2>}"`

  For convenience, set the cts_annotated_delay* attribute using a single value that will apply to all available delays from pins and rise/fall from edges. For example, "0.100".
- `cts_annotated_transition(pin)`: Overrides any clock tree timing engine computed transition at this pin, in a similar manner to SDC set_annotated_transition.
  Default: `auto`
- `cts_annotated_transition(port)`: Overrides any clock tree timing engine computed transition at this pin, in a similar manner to SDC set_annotated_transition.
  Default: `auto`

Note: For details about the above attributes, see *Stylus Common UI Database Object Information* manual.

Parameters

`-help`	Outputs a brief description of the `update_clock_tree_spec_annotations` command. For a detailed description of the command, use the man command: `man update_clock_tree_spec_annotations`.
`-out_file file_name`	Writes commands to the specified file instead of applying the changes in memory.

Example

Use the set of commands listed below to define a transition annotation at `pin b1/I`, delay annotation at `net_b1`, update the annotations from CTE to clock tree timing engine (CTTE), and verify that the annotation is corrected applied at clock tree timing engine (CTTE).

```
set_interactive_constraint_modes [all_constraint_modes]

set_propagated_clock [all_clocks]

create_ccopt_clock_tree -name clk -source clk -no_skew_group
create_skew_group -name clk/default_constraint_mode -sources clk -auto_sinks
```

Annotate transition at b1/I pin using the below command:

```
set_annotated_transition 1.234 [get_pins {b1/I}]
```

Annotate delay at net_b1 net using the below command:

```
set_annotated_delay 2.345 -net -from [get_pins {b1/Z}] -to [get_pins {b2_2/I}]
```

Update the annotations from CTE to CTTE using the below command:

```
update_clock_tree_spec_annotations
```

The log shows the following information:

```
<CMD> update_clock_tree_spec_annotations
Ignoring AAE DB Resetting ...
Updating timing graph...

Leaving CCOpt scope - BuildTimeGraph...
AAE DB initialization (MEM=1059.9 CPU=0:00:00.2 REAL=0:00:00.0)
################################################################################
# Design Stage: PreRoute
# Design Name: top
# Design Mode: 90nm
# Analysis Mode: MMMC Non-OCV
# Parasitics Mode: No SPEF/RCDB
# Signoff Settings: SI Off
################################################################################
Calculate delays in Single mode...
Start delay calculation (fullDC) (1 T). (MEM=1067.43)
siFlow : Timing analysis mode is single, using late cdB files
Total number of fetched objects 8
AAE_INFO: Total number of nets for which stage creation was skipped for all views 0
End delay calculation. (MEM=1141.94 CPU=0:00:00.0 REAL=0:00:00.0)
End delay calculation (fullDC). (MEM=1130.4 CPU=0:00:00.4 REAL=0:00:01.0)
Leaving CCOpt scope - BuildTimeGraph done. (took cpu=0:00:00.7 real=0:00:00.9)
Updating timing graph done.
Updating latch analysis...
Leaving CCOpt scope - Updating latch analysis...
Leaving CCOpt scope - Updating latch analysis done. (took cpu=0:00:00.0 real=0:00:00.0)
Updating latch analysis done.
Clock tree spec update: Transition and delay annotations were updated to match active timing constraints.
```

Verify that the annotations are correctly updated using the below commands:

```
get_clock_tree_transition_time -pin b1/I
<CMD> get_ccopt_clock_tree_transition_time -pin b1/I
Clock tree timing engine global stage delay update for default_delay_corner_max:setup.late...
Clock tree timing engine global stage delay update for default_delay_corner_max:setup.late done. (took
cpu=0:00:00.2 real=0:00:00.2)
1.234
```

Related Commands

- get_clock_tree_transition_time
- create_clock_tree
- create_skew_group
- get_pins
- set_annotated_delay
- set_annotated_transition
- set_interactive_constraint_modes
- set_propagated_clock

update_skew_group

```
update_skew_group
[-help]
[-make_exclusive]
-skew_group skew_group_name
[-add_sinks pins | -remove_sinks pins]
[-add_ignore_pins pins | -remove_ignore_pins pins]
```

This command is used internally to add or remove sinks in existing skew groups produced automatically by the `create_skew_group` command, but you can also use it to modify existing skew_group objects. Sinks and ignore pins can be added or removed using the appropriate switches.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `update_skew_group` parameter. For a detailed description of the command and all of its parameters, use the man command: `man update_skew_group`.
`-add_ignore_pins pins`	Specifies the ignore pins to be added to the skew group. *Data_type*: string, optional
`-add_sinks pins`	Specifies the sink pins to be added to the skew group. *Data_type*: string, optional
`-make_exclusive`	When specified, sets the exclusive sink rank of the specified skew group, using the `-skew_group` parameter, to be the highest one. *Data_type*: bool, optional
`-remove_ignore_pins pins`	Specifies the ignore pins to be removed from the skew group. *Data_type*: string, optional
`-remove_sinks pins`	Specifies the sink pins to be removed from the skew group. *Data_type*: string, optional
`-skew_group skew_group_name`	Specifies the user-defined identifier for the skew group to be modified. *Data_type*: string, required

Example

- The following command specifies the sink pins to be added to the specified skew group:
  ```
  update_skew_group -skew_group clkA -add_sinks f1/ck
  ```

Related Information

- `create_skew_group`
- `get_skew_group_path`
- `get_skew_group_delay`
- `delete_skew_groups`
- `report_skew_groups`
- The following sections in Clock Tree Synthesis chapter of *Innovus Stylus Common UI User Guide*
  - Automatic Clock Tree Specification Creation
  - Manual Setup and Adjustment of the Clock Specification

write_cts_config

```
write_cts_config
[-help]
out_dir
```

Writes the CTS configuration to the specified directory.

Note: The content of the files generated using this command is proprietary. The files may not be in any particular format, or even human-readable, and are subject to change. The only supported use of the output of `write_cts_config` command is as input to the `read_cts_config` command.

Parameter

`-help`	Outputs a brief description of the `write_cts_config` parameter.
out_dir	Specifies the directory path to which the CTS configuration will be written. *Data_type*: string, required

Example

- The following command writes the CTS configuration to the specified directory:
  ```
  write_cts_config ../ccopt_tutorial/work
  ```

Related Commands

- `check_cts_config`
- `reset_cts_config`
- `read_cts_config`

7

Conformal Commands

- run_clp

run_clp

```
run_clp
[-cpf fileName]
[-cmd cmdFile]
[-extra_libraries lib …]
[-extra_verilog fileName …]
[-setup_only]
[-use_eeq_cell_with_liberty_info]
```

Generates the following files, then runs the Conformal® Low Power Verification tool, or generates a template script to run the Conformal® Low Power Verification tool the ConformalÃ, ® Low Power Verification tool.

- Netlist (`clp_input/feclp.v`)
- CPF/1801 file (`feclp.cpf or feclp.upf`)
- dofile (`clp_input/feclp.tcl`)

Parameters

`-extra_libraries` *lib …*	Specifies a list of extra libraries (`-liberty`). *Data_type*: string, optional
`-extra_verilog` *fileName …*	Specifies a list of extra Verilog files (`-verilog`). *Data_type*: string, optional
`-cpf` *fileName*	Specifies the CPF file to use as input to the verification tool. *Default:* `feclp.cpf`, generated by the Innovus software. *Data_type*: string, optional
`-cmd` *cmdFile*	Specifies the Tcl file to use as input to the verification tool. *Default*: feclp.tcl, generated by the Innovus software. *Data_type*: string, optional
`-setup_only`	Generates input files, but does not run the verification tool. *Data_type*: bool, optional
`-use_eeq_cell_with_liberty_info`	Uses an EEQ cell with timing library definition as the master cell of a leaf instance. *Data_type*: bool, optional

Example

The following command will write out template CLP script, clp_input/feclp.tcl:

```
> run_clp -setup
```

8

Database Access and Handling Commands and Attributes

- define_attribute
- delete_buffer
- delete_obj
- get_db
- get_db Category Attributes
- is_attribute
- reset_db
- set_db
- sort_db
- write_preserves

define_attribute

```
define_attribute
[-help]
attribute_name
-obj_type object_type
[-obsolete]
{-alias existing_attribute | {-category category [-data_type {bool | int | double | area | coord | string | enum | point | rect | line |
delay | resistance | capacitance | voltage | in_dir | out_dir | in_file | out_file  | object | object*} [-enum_values values]]
[-check_function check_function]
[-compute_function compute_function]
[-default default]
[-help_string description]
[-hidden]
[-set_function set_function]
[-skip_in_db]}}
```

Creates a new, user-defined attribute with the specified characteristics. The attribute created is a first-class db object whose name is `object_type/attribute_name`. Use `delete_obj` to undo `define_attribute`.

The attribute value can be reset whether or not the default value is specified. If a default value is specified, `reset_db` will use it. If no default value is specified, `reset_db` resets the default value to one of the following:

- `no_value` for simple numeric types such as `int`, `double`, `delay`, `capacitance`, `coord`, `area`, `resistance`, and `voltage`
- `false` for Boolean types
- "" (empty string) for all other types, including `string`, `in_file`, `out_file`, `in_dir`, `out_dir`, `point`, `rect`, and `line`

Note that the new user-defined attribute created by `define_attribute`, and the attribute value can be recorded in the def file using the `write_def` command only after resetting the value using the `set_db` command, The default user-defined attribute and its value cannot be recorded in the def file using the `write_def` command.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `define_attribute` parameter. For a detailed description of the command and all its parameters, use the man command: `man define_attribute`

`-alias existing_attribute`	Makes *attribute_name* a hidden alias for *existing_attribute*, which must have been defined. This parameter is used along with the `-obj_type` parameter. It could also be used with the `-obsolete` parameter to generate an obsolete warning. *Data_type*: string, optional
attribute_name	Specifies the name of the attribute to be defined. Note that the name of the underlying db object that is created is `object_type/attribute_name`. *Data_type*: string, required
`-category category`	Specifies the category of the attribute. Categories group attributes that perform similar functions, whereas `object_type` specifies on which type of object the attribute is valid. You can specify any category name, both new and existing category names are allowed. *Data_type*: string, required
`-check_function check_function`	Specifies a previously defined Tcl procedure's name to ensure that the newly defined attribute is valid. The Tcl procedure should be of the form `proc {object value} { <body returns 0 or 1> }`. *Data_type*: string, optional
`-compute_function compute_function`	Specifies a previously defined Tcl procedure's name to get the newly defined attribute's value later (with the `get_db` command). The Tcl procedure must be of the form `proc {object} { <body returns value of type data_type> }`. When you use this option, the attribute becomes a read-only attribute because its value is always computed. *Data_type*: string, optional
`-data_type {bool | int | double | area | coord | string | enum | point | rect | line | delay | resistance | capacitance | voltage | in_dir | out_dir | in_file | out_file | object | object*}`	
	Defines the data type of the attribute. Possible data types are: bool, int, point, string, etc., as described in the Types and Definitions chapter in the *Stylus Common UI DB Object Information* manual. The `object` and `object*` data types correspond to `obj` and `obj*` in the *Stylus Common UI DB Object Information* manual. Only those objects that have a DPO with a unique name (like `inst:top/i1`) are supported. Objects without a DPO name (like `wire:0x222222222`) are not supported. *Data_type*: enum, required
`-default default`	Specifies a default value for the attribute. In case this parameter is not specified while creating an attribute, the default value will be: • `no_value` for simple numeric types such as `int`, `double`, `delay`, `capacitance`, and `coord` • `false` for Boolean types • "" (empty string) for all other types, including `enum`, `string`, `in_file`, `out_file`, `in_dir`, `out_dir`, `point`, `rect`, `polygon`, `line`, and `obj_type` *Data_type*: string, optional
`-enum_values values`	Specifies a list of all the possible values of the enumerated type. For example, `-enum_values {a b c}`. *Data_type*: string, optional
`-help_string description`	Specifies the help text for the attribute. *Data_type*: string, optional
`-hidden`	Specifies whether the defined attribute is a hidden attribute. *Data_type*: bool, optional
`-obj_type object_type`	Specifies the object type of the attribute. All valid object types can be found by running the following command through the command prompt: `get_db obj_types -if {.accept_user_defined_attributes}` The attribute is assigned to the specified type. *Data_type*: string, required
`-obsolete`	Specifies if the defined attribute is obsolete, and gives a warning if used. *Data_type*: bool, optional

`-set_function` *set_function*	Specifies a previously defined Tcl procedure's name. This option allows you to override user-defined values provided it conforms to the parameters in the Tcl procedure you created. The Tcl procedure should be of the form `proc {object new_value current_value} { <body returns value of type data_type> }`. Returns new value to be assigned to the attribute. *Data_type*: string, optional
`-skip_in_db`	Prevents the `write_db` command to write out the defined attribute in the db being written. *Data_type*: bool, optional

Examples

- The following command creates a Tcl procedure, `check_fxn`, and then creates an attribute named test_check. The `-check_function` parameter is specified so that the test_check attribute can be tested for validity when it is specified later:

  ```
  proc check_fxn {obj val} { if { $val > 0 } { return 0}; r eturn 1 }
  define_attribute test_check1 -obj_type inst -data_type int -category test -help_string "test_check_function" -check_function  check_fxn
  ```

  Based on the check_function parameter, the checking criteria attribute value can be set only to a negative value. As a result the following command sets the attribute value to -1:

  ```
  set_db insts .test_check1 -1
  ```

  but the following command displays an error and keeps the previous value.

  ```
  set_db insts .test_check1 2
  ```

- The following command creates a Tcl procedure, `set_fxn`, and then creates an attribute named `test_set`. The `-set_function` parameter is specified so that you can change the value of the `test_set` attribute (provided it is valid):

  ```
  proc set_fxn {obj new_val cur_val} { if {$new_val > $cur_val} { return $new_val }; r eturn $cur_val }
  define_attribute test_set -obj_type root -data_type int -category test -help_string "test set function" -set_function set_fxn -default 0
  ```

  Now, the following command sets the attribute value to 1 because the default value is 0

  ```
  set_db test_set 1
  ```

  The following command changes the attribute value to 4 as the new value is greater than the current value

  ```
  set_db test_set 4
  ```

  But with the following command the value is not changed as the new value is less than the current value (4)

  ```
  set_db test_set 2
  ```

- The following command uses the `compute_function` to set attribute values. Note that the attribute becomes read-only if `compute_function` is applied.

  ```
  proc comp_func {obj} { return 5 }
  define_attribute test_compute -obj_type root -data_type int -category test -help_string "test set function" -compute_function comp_func
  get_db test_compute 5
  ```

  Now, the following command will issue an error as attribute value cannot be changed.

  ```
  set_db test_compute 3
  ```

- The following command creates an alias to a pre-existing attribute:

  ```
  define_attribute attr1 -alias old_hidden_attr2 -obj_type root
  ```

  Note that the old_hidden_attr2 attribute must be pre-defined, else the above command will return an error message. Note that in general define_attribute creates an attribute db object–but when invoked with the -alias switch it doesn't.

- The following commands check to see if the attribute exists and if not creates the attribute, then deletes the attribute:

  ```
  if { ! [ is_attribute myAttribute -obj_type inst ] } { define_attribute myAttribute -obj_type inst -data_type bool -category
  myAttributes }
  delete_obj [ get_db attributes inst/myAttribute ]
  ```

- The following commands define new attribute of the `design obj_type`, and then record the attribute and its current value, which is reset by the user through the `set_db` command:

  ```
  define_attribute test_1 -data_type string -obj_type design -category write_def -help_string {DEF property} -default_value {abc} -quiet
  set_db designs .test_1 aabbcc
  write_def new.def

  #new.def content#
  ```

```
PROPERTYDEFINITIONS
DESIGN test_1 STRING "aabbcc" ;
```

Related Topics

- For information on how to query DB attributes, see the `get_db` command.
- For information on how to set DB attributes, see the `set_db` command.
- For information on pre-defining `check/set/compute` functions for user defined attributes embedded in saved data-bases, see the read_db command
- For information on how to delete data base objects (including attributes), see the delete_obj command
- Use is_attribute to determine if an attribute has already been created

delete_buffer

```
delete_buffer
[-help]
-buffer buffer_name
```

Deletes the specified buffer for a list of instances.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `delete_buffer` parameter. For a detailed description of the command and all its parameters, use the man command: `man delete_buffer`
`-buffer buffer_name`	Specifies the list of buffers to be deleted. *Data_type*: (inst)+, required

Examples

Following are few examples of the command usage for removing the specified buffers:

```
delete_buffer -buffer b1
delete_buffer -buffer u1/b1
delete_buffer -buffer {u1/b1, u2/b1} # u1 and u2 are master-clone hierarchical instances
delete_buffer -buffer {u1/b1 u2/b2 b3}
```

delete_obj

```
delete_obj any_object+
[-help]
```

Deletes one or more existing objects from the database. The command can delete `netlist` objects and physical data like `routing` and `floorplan` objects, but not the `clock` or `timing` objects (such as `clock_tree`, and `rc_corner`).

The following `obj_types` can be deleted through this command: `attribute`, `bump`, `bus_guide`, `flow`, `flow_step`, `group`, `gui_text`, `gui_line`, `gui_rect`, `gui_polygon`, `hinst`, `hnet`, `hpin`, `hport`, `inst`, `marker`, `module`, `net`, `net_group`, `patch_wire`, `pin_guide`, `pin_group`, `pin_blockage`, `place_blockage`, `port`, `port_shape`, `resize_blockage`, `route_blockage`, `route_type`, `row`, `sdp`, `special_wire`, `special_via`, `text`, `via`, `virtual_wire`, `what_if_via`, `what_if_wire`, and `wire`.

Parameters

`-help`	Provides a brief description of the command that includes the type and default information for parameters. For a detailed description of the comm `man delete_obj`
`any_object+`	Specifies the object or the list of objects to be deleted. *Data_type*: (special_wire|special_via|place_blockage|route_blockage|pin_blockage|resize_blockage|text|gui_text|gui_line|gui_rect|gui_polygc flow|flow_step|wire|via|patch_wire|virtual_wire|port_shape|attribute|route_type|what_if_wire|what_if_via|hinst|hpin|hport|port|module|inst|net|hr

Examples

- The following command deletes all the `place_blockages`:
  ```
  delete_obj [get_db place_blockages]
  ```
- The following command deletes the specified `place_blockage` object:
  ```
  delete_obj place_blockage:0x2b77dfbf0048
  ```
- The following command deletes all the special routes from the VDD net:
  ```
  set my_net [get_db pg_nets VDD]
  delete_obj [get_db $my_net .special_wires]
  delete_obj [get_db $my_net .special_vias]
  ```
- The following command deletes a group of SDPs:
  ```
  delete_obj sdp:DTMF_CHIP/wCardInstGrp_1
  ```

Related Information

- For information on querying the database objects, see the `get_db` command.
- For a list of objects that can be queried for information or changed, see the Common UI Database Object Information document.

get_db

```
get_db
[-help]
{[start [chain] [pattern1 [pattern2 ..] [[-if expression] | [-expr expression]] [-dbu] [-foreach tcl_body] [-index {object | obj_type1
name1 [object | obj_type2 name2 …]}] [-invert] [-regexp] [-match_hier] [-unique] [-computed]}
[-category category]
[-depth {[min] max}]
[-skip_fail]
```

Returns objects and attributes in the database with different types of filtering. The `get_db` commands takes a dual-ported object, dual-ported object list or a collection as input, then uses a period (.) to traverse to other related objects or to access attributes. You can use additional periods to traverse the data base schema.

Note: A Dual-Ported Object (DPO) is a Tcl_Obj in the C++ programming interface. The pointer is kept as a C++ pointer inside Tcl unless Tcl forces it to be evaluated as a string. In normal usage it never gets converted to a string which is more efficient, but if you do a "puts $var" or return the value to the shell, it will be converted to its "dual" string form.

- The string form is normally a useful name preceded by the `obj_type`, so a `base_cell` object string form might look like `base_cell:and2`.
- A netlist name will have the current design name included, so an instance named `i1/i2` in design top, would look like `inst:top/i1/i2`.
- An object with no name like a wire, will just show the pointer hex-value like `wire:0x22222`.

The `get_db` command accepts collections as an input but it only returns objects in dual-ported object list form.

For descriptions on all objects and attributes, see Common UI Database Object Information.

The root object has the current design data under the `current_design` attribute, while technology and library data are normally directly accessible as root attributes (e.g. layers and base_cells). The `root` object is the default starting point if there is no input list or collection given.

You can query a root attribute in any of the following ways:

- `get_db / .layers`
  where, '/' is the notation for the root object
- `get_db layers`
  This implicitly starts at the root. In this usage, no "." is used before the "layers" attribute name.

For example, you can use the following command to get all the driver-pins of all nets in `current_design`:

```
>get_db current_design .nets.drivers
```

Note: The `current_design` is implicitly created by the `init_design` or the `read_db` commands.

Frequently used current_design attributes like insts or hinsts (hierarchical insts) have root attribute aliases (or shortcuts). These attributes can be accessed by just typing the root attribute alias name.

For example, the following commands gets all instances of the current design with names that start with `i1/i2*`:

- `get_db current_design .insts i1/i2*`
- `get_db insts i1/i2*`

Where, `insts` is a root attribute alias for `current_design .insts`

Some of the more commonly used aliases from the `current_design` attributes include: `nets`, `pins`, `insts`, `hnets`, `hpins`, and `hinsts`.

The `get_db` command supports auto-complete when you press the <`Tab`> key. It will give a list of all matching attributes starting with the letters entered so far, or it will complete the name if it is unique.

For example:

```
>get_db insts .par<Tab>

parent partition

>get_db insts -if {.par<Tab>

parent partition
```

If you press the `<Shift+Tab>` keys together, the matching attributes are displayed along with their help string, like this:

```
>get_db insts .par<Shift+Tab>

parent # The parent of the inst, which is either a hinst for an instance within hierarchy or the design object for an instance at
the toplevel.
partition # If this inst is a physical black_box, this partition will carry the pin constraints and related data for the blackbox.
Otherwise it is empty.
```

There are two database definition objects: `obj_type` and `attribute`. If you want to query all the details for an attribute definition on an `obj_type`, rather than an attribute value, use the following command:

```
get_db attribute:<obj_type>/<attribute_name> .*
```

You can use `get_db` to query user-defined properties from LEF. Following is the mapping between the LEF `objectType` and database object:

```
LAYER -> layer
LIBRARY -> design
MACRO -> base_cell
NONDEFAULTRULE -> route_rule
PIN -> base_pin
VIA -> via_def
VIARULE -> via_def_rule
```

Note: If `PROPERTYDEFINITION` is defined for `LIBRARY`, the `PROPERTY` will be mapped as a design attribute with a `tech_lef_property_*` prefix.

Copying and pasting pointers in an active session may cause the tool to crash. To prevent this situation, do not:

- Copy pointers from the output of the `get_db` command and paste to another command as input.
- Copy a pointer from one session to another.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `get_db` parameter.

`-category` *category*	Specifies a category of root attributes to return. This is intended to return all the root attributes associated with one application c used for anything but root attributes. The return value is list of root attribute names.
	Example:
	The following command can be used to see all the root attributes associated with the category place:
	`get_db -category place`
	The output is as follows:
	`place_design_floorplan_mode place_design_refine_place place_detail_allow_border_pin_abut ...`
	You can view the complete list of categories with the get_db categories command as follows:
	`innovus 1> get_db categories`
	`active_logic_view add_endcaps add_fillers add_rings ...`
	Data_type: string, optional

chain	Specifies the *chain* in the form `[.attribute_name1 [.attribute_name2]...]`. A chain always starts with a `"."`.
	The specified attribute names are valid for each level of the traversal in the *chain*.
	Example:
	The following command returns the names of pins of `$my_insts`
	`get_db $my_insts .pins.name`
	If the chain has a * or ? in the attribute name, it will be expanded to all matching attribute names, and the display will show the a values are returned to Tcl. Values that may require significant time to compute (like timing data that requires the timing-graph) v
	For example:
	`get_db $my_insts .b*`
	`Object: inst:top/i1`
	`base_cell: base_cell:BUFX2`
	`bbox: {34.32 10.08 38.94 15.12}`
	`Object: inst:top/i2`
	`...`
	Data_type: string, optional

`-computed`	Forces `get_db` to output the values of attributes that may require significant time to compute (like timing data that requires the ti usage when using a pattern match with * or ? for an attribute name. If only a single attribute name is given, then the value is alw
	Example:
	`>get_db $my_pins .slack_max*`
	`Object: pin:top/inst$odd/Y`
	`slack_max: NC`
	`slack_max_edge: NC`
	`...`
	`>get_db $my_pins .slack_max* -computed #if timing-graph not available yet, it will be computed`
	`Object: pin:top/inst$odd/Y`
	`slack_max: 550.12`
	`slack_max_edge: rise`
	`...`
	`>get_db $my_pins .slack_max #single attribute names always return value and force computation if needed`
	`550.12 ...`
	Data_type: bool, optional

`-dbu`	Indicates that attribute values (coord, pt, rect, polygon, area, ...) should be returned as an int in database units.
	Default: If not given, the value returned is a double in microns (or microns**2).
	Examples:
	• In the following command the return type is a double in microns.
	`get_db $my_inst .location`
	`{1.0 1.0}`
	• In the following command the return type is an int in database units.
	`get_db -dbu $my_inst .location`
	`{1000 1000}`
	Data_type: bool, optional

`-depth {[min] max}`	Specifies the hierarchy depth range. It expands each `design` or `hinst` object in the input list into a list of `design` or `hinst` objects specified. `0` here signifies the level of the starting object. The `-depth` expansion happens first, and the resulting expanded list is then processed as the input list of objects by `get_db`. For example: `get_db hinst:top/h1 -depth {0 1}` Returns `hinst:top/h1`, and the first level of hinsts below it. *Data_type*: string, optional

`-expr` *expression*	Returns the object if the *expression* is true. The *expression* argument is a strict Tcl expression that can be passed as is to the with any attribute or chain value for each object as they are visited, and `$obj(.)` returns the object itself. For example,
	`get_db insts -expr {$obj(.base_cell.name) eq "AND2"}  #all insts that are AND2`
	Normally the `-if` *expression* parameter is faster for simple expressions, but it is limited to a few operators, and simple `$var` sul char escaping, or complex `$var` substitution, you must use `-expr`.
	This parameter can be used with a *chain* and `pattern` like this:
	`get_db $my_insts .pins i1/* -expr {$obj(.base_name) eq "A"}`
	Here, the `i1/*` pattern is checked first for pin names that match `i1/*`, then the pin `.base_name` is checked if it is equals "`A`".
	`-expr` cannot be used at the same time with `-if`.
	Data_type: string, optional

`-foreach` *`tcl_body`*	Specifies a list of Tcl commands to process for each object. This parameter avoids the overhead of creating long lists of objects `$obj(.)` is the object itself. The Tcl commands break and continue also work as expected in a `foreach` loop (break stops the tra Unlike inside `-if`, the `tcl_body` is strict Tcl. The *`.attribute`* names are not substituted, you must use `$obj()` to access the attrib each `object` that matches the *`pattern`*, `-if`, or `-expr` expression. The following command prints the names of every inst that starts with "abc" and is an AND2 gate: `get_db insts abc* -if {.base_cell.name == AND2} -foreach {puts $obj(.name)}` It returns the total number of `insts` that are printed. *Data_type*: string, optional

`-if` *`expression`*	Specifies a boolean expression, with extensions to make pattern matching simpler.
	• == and != do not require a quoted string, and they support pattern matching with * and ?.
	• Other operators like >, <, <=, >=, &&, ||, and the unary negate operator ! are supported.
	• Parentheses "( )" can be used to group complex expressions.
	
	Note: Unlike the <pattern> argument, inside the -if expression, you cannot use a string pattern to implicitly match the .name attr will not match anything because .base_cell will become something like `base_cell:AND2`.
	
	`>get_db insts -if {.base_cell == buf*} #matches nothing`
	
	use either of these instead:
	
	>get_db insts -if {.base_cell.name == buf*} #match the .name attribute
	
	>get_db insts -if {.base_cell == base_cell:buf*} #match the DPO name directly
	
	Note: If you use an attribute that is an object list (like .pins), every object in the list is tested in an "implicit OR", so if any of them
	
	Handling of indexed attributes inside of the `-if` statement:
	• The `-index` option is kept outside of the `-if` statement
	• The `-index` option MUST be valid for ALL attributes inside the `-if` statement otherwise an error occurs
	
	Examples:
	• The following command finds all instances with `base_cell` names that matches "`buf*`".
	`get_db insts -if {.base_cell.name == buf*}`
	• The following command returns all insts that have one or more pins that match `CLK*`. Only one copy of the matching inst obje
	`get_db insts -if {.pins.base_name == CLK*}`
	• The following command gets all pins where `delay_max_rise` is greater than 5 or `slew_max_fall` is smaller than 8. This is a v abbreviation of `analysis_view`).
	`get_db pins -if {.delay_max_rise > 5 || .slew_max_fall < 8 } -index {view v1}`
	
	Note: The `-if` parameter does not support the usage of embedded Tcl command as operands. Use the `-expr` for this requirem
	
	Data_type: string, optional

`-index {object | obj_type1 name1 [object | obj_type2 name2 …]}`	Some attributes represent an array of values. For example, a `pin .slack_max_rise` attribute has a different value for each `anal` attribute is used (for example, the maximum, or minimum of all values in the array). For `slack_max_rise` with no `-index` value, th
	If you want to return a specific value, you must include the index value. The index values are object pointers, or a pair of `obj_ty` abbreviation for `analysis_view`.
	For example:
	`get_db $my_pins .slack_max_rise -index "view func_wc_cworst"`
	Will return the worst case `.slack_max_rise` for the `analysis_view func_wc_cworst` across all clocks. This attribute also accepts
	`get_db $my_pins .slack_max_rise -index "view func_wc_cworst clock clk1"`
	To get the value for that specific `analysis_view` and `clock`.
	In the special case of a clock and `analysis_view` as shown above, a clock object is sufficient for both, because a clock object is
	`set my_clk_obj [get_db clocks -if {.view_name == func_wc_worst && .base_name == clk1}]`
	`get_db $my_pins .slack_max_rise -index $my_clk_obj`
	The `-index` parameter also accepts a pattern of attribute name to be matched. While specifying an attribute pattern string with `-` support `-index` will return the values specified by the index. For example:
	`get_db net:dtmf_recvr_core/scan_out2 .* -index {view func-wc_cworst}`
	Will return all the attributes and values for the `net` object. Any attribute that supports a view index will return the value for that vi
	Data_type: tcl_obj, optional

`-invert`	Negates the *`pattern`*, `-if` *`expression`*, `and -expr` *`expression`* filters to return the objects/attributes that do not match instead *Data_type*: bool, optional
`-match_hier`	Specifies that the `*` in the *`pattern`* and *`expression`* expands to the next hierarchical divider character `/` instead of a complete str Example: The following command matches i1, i2, but not i1/i2. `get_db insts i* -match_hier` *Data_type*: bool, optional

pattern1 [pattern2 ..]	Specifies string pattern(s), where:
	* matches 0 or more of any characters
	? matches exactly one character
	More complex patterns require `-regexp` or `-if` expression.
	Refer to the following examples:
	• The following command finds all instance names starting with A:
	`get_db insts .name A*`
	`A0 A1/A2 …`
	• The pattern matching can also work directly on a dual-ported object. In that case, it will match the '.name' attribute of the obj have a `.name` attribute defined. For example, the following command finds all instance objects (not names) with a name star
	`get_db insts A*`
	`inst:top/A0 inst:top/A1/A2 …`
	• The pattern matching can also work to match multiple lists of the name patterns. Refer to the following example:
	`get_db {*/BC* *TC*}`
	`CG/BC1 CG/BC2 TC1 TC2 TC3 TC4`
	`0x19f`
	Note: For the special case where a pattern string starts with '.' you need to use an extra '.' (an empty attribute name) so that '.text' is treated as a pattern and not as an attribute name.
	• The following command finds all nets whose names equal to `.odd_name`
	`get_db nets . .odd_name`
	• The following command finds all instance names starting with A or B:
	`get_db insts A* B*`
	Data_type: string, optional

`-regexp`	Specifies that *pattern* use *regexp* syntax instead of the simple * and ? style. It does not change the `-if` pattern matching. ⚠ This parameter matches the specified pattern with the whole reference string. See the example below to understand the usage of the `-regexp` parameter: `get_db insts -regexp {u_warmreset_cntr/reset_cntr_reg_(\d+)_}` *Data_type*: bool, optional
`-skip_fail`	Skips displaying the fail error, when the object does not have the specified attribute. *Data_type*: bool, optional

start	The starting objects or root attributes. Can be an object list, collection, root (/), or root attribute.
	Example:
	• The following command returns the root attribute `insts` that is an object list of all `insts` in `current_design`.
	`>get_db insts`
	• The following command starts from a dual-ported object list or collection of objects in the Tcl variable `$my_list`, and returns
	`>get_db $my_list`
	• The following command returns a root attribute that is a simple value.
	`>get_db timing_cap_unit`
	• If the name has a * or ?, it will be expanded to all matching root attribute names, and the display will show the attribute name example,
	`>get_db timing*`
	`Object: root:/`
	`timing_allow_input_delay_on_clock_source: false`
	`timing_analysis_aocv: false`
	`...`
	Data_type: tcl_obj, required

`-unique`	Removes duplicate entries from the return list. If the specified attribute is a list, this parameter returns a unique set of lists., and *Data_type*: bool, optional

Built-In Sub-attributes

The types `rect`, `line`, `point`, and `polygon` have built-in sub-attributes. For a full list of all built-in sub-attributes, see Common UI Database Object Information.

Example:

The `rect` type has built-in sub-attributes like:

area	Area of rect
dx	Delta x
dy	Delta y
ll	Lower-left point
ur	Upper-right point
width	Shorter of dx and dy
length	Longer of dx and dy
perimeter	Length of the rect perimeter

A `point` has sub-attributes like:

x	X coordinate
y	Y coordinate

Like other attributes you can chain them together.

For example, an inst object has a .bbox (bounding box) that is type “rect”. Here are some operations allowed on the .bbox attribute of an inst.

```
get_db $my_inst .bbox

{1.0 2.0 3.0 4.0}

get_db $my_inst .bbox.ll

{1.0 2.0}

get_db $my_inst .bbox.area

4.0

get_db $my_inst .bbox.ll.x

1.0

set box [get_db $my_inst .bbox]

{1.0 2.0 3.0 4.0}

get_db $box .ll

{1.0 2.0}
```

Examples

- The following command finds insts with names starting with PTN* that have at least one pin capacitance > 0.1. The pattern is checked first, and those that pass will be tested with the `-if` expression. Inside the `-if` expression, there are many `.pins` for one inst, so each pin is tested. If any of the pins have a capacitance > 0.1, the single inst will be matched.
  ```
  get_db insts PTN2* -if {.pins.capacitance_max_rise  > 0.1}
  ```
- The following command returns insts that are not physical-only cells whose name matches `i1/i2*`.
  ```
  get_db insts i1/i2* -if {!.is_physical}
  ```
- The two commands below generate the same list of pins from `$my_insts` that have a base_name `clk`. The first uses a * pattern to match all pins, while the second has no pattern and defaults to all pins, so both cases check the `-if` expression for all pins of `$my_insts`.
  ```
  get_db $my_insts .pins * -if {.base_name == clk}
  ```

```
get_db $my_insts .pins  -if {.base_name == clk}
```

- The following command finds `buf1` cell instances connected to nets named `clk` and changes their placement status to `fixed`.
  Note: The examples below is inefficient and used to illustrate different methods of traversal done in different styles. A more efficient method would be to start from the nets and looking at their pin connections directly.
  Example 1:

```
set cell_name buf1
set net_name clk
foreach inst_obj [get_db insts] {
      set cell_obj [get_db $inst_obj .base_cell]
      if { [get_db $cell_obj .name] == $cell_name } {
               foreach pin_obj [get_db $inst_obj .pins] {
                     if { [get_db $pin_obj .net.name] == $net_name } {
                         set_db $inst_obj .place_status fixed
                     }
               }
       }
}
```

  Example 2: Same output using full traversal style with intermediate variables. The second `-if` returns true if any of the .pins.net.name match `$net_name` (implicit OR).

```
set buf1_list [get_db insts -if {.base_cell.name == $cell_name}]
set buf1_clk1_list [get_db $buf1_list -if {.pins.net.name == $net_name}]
set_db $buf1_clk1_list .place_status fixed
```

  Example 3: Same output using full traversal style (fully chained)

```
set_db [get_db insts -if {.base_cell.name == $cell_name && .pins.net.name == $net_name}] .place_status fixed
```

  Example 4: Or use the `-foreach` method so no list is returned, but the number of buffers processed is returned.

```
get_db insts -if {.base_cell.name == $cell_name && .pins.net.name == $net_name} -foreach {set_db $obj(.) .place_status fixed}
```

- The following command finds all the insts named `i1/i2` that end with `_buf`:

```
get_db insts i1/i2/*_buf
```

  Or your could start inside the `hinst` named `i1/i2`:

```
get_db [get_db hinsts i1/i2] .insts *_buf
```

  Or use the hinst dual-ported object name directly. Note that netlist dual-ported object names have the design name included, so if `top` is the design name you could do this:

```
get_db hinst:top/i1/i2 .insts *_buf
```

- The following command gets `arrival_max_rise` for the `analysis_view setup_test` for all pins. Here the abbreviation "view" is allowed for analysis_view.

```
get_db pins .arrival_max_rise -index "view setup_test"
```

- The following command finds insts that have a cell_port named CK and that pin is connected to a net named clk*:

```
get_db insts -if {.lib_cells.lib_pins.name == */CK && .pins.net.name == clk*}
```

- The following command uses -foreach to find the first inst named u1/* with .place_status "fixed" and then break. It uses strict Tcl inside the -foreach.

```
get_db insts u1/* -foreach { if {[string match fixed $obj(.place_status)]} { puts $obj(.name) ; break } }
```

- The following `get_db` command uses `-foreach` to count the number of each base_cell used in the netlist inside the cell_count Tcl array. You can then use `array get cell_count` to see a list of all the `base_cell` names and how many times they are used in the netlist.

```
get_db insts .base_cell -foreach {incr cell_count($obj(.name))}
```

- The following `get_db` command displays all the details of the `flow_template_type` attribute of the `root obj_type`:

```
get_db attribute:root/flow_template_type .*
Object: attribute:root/flow_template_type
base_name: flow_template_type
category: flow
check_function: {}
compute_function: {}
data_type: string
default_value: {}
help: {Type of template being run.}
indices:
is_computed: false
is_obsolete: false
```

```
is_settable: true
is_user_defined: true
name: root/flow_template_type
obj_type: obj_type:root
set_function: {}
skip_in_db: false
```

- The following example depicts the use of the `-if` parameter to find all HVT insts:
  ```
  get_db insts -if {.base_cell.name == *_HVT}
  ```
- The following command returns all hinsts 1 or 2 levels inside the design:
  ```
  get_db design:top -depth {1 2}
  hinst:top/h1 hinst:top/h1/h2
  ```
- The following command finds all `local_insts` of each `hinst` at depth 2 that have `.dont_touch == true`:
  ```
  get_db design:top .local_insts -depth {2 2} -if { .dont_touch == true }
  inst:top/h1/h2/i1 inst:top/h1/h2/i2
  ```
- All the following commands return inst with `.name i1/i2`:
  ```
  set a i1/i2
  get_db insts i1/i2
  get_db insts -expr {$obj(.name) eq $a}
  get_db insts -expr {[get_db $obj(.) .name] eq "i1/i2"}
  ```
- The following command returns all nets with more than 5 loads:
  ```
  get_db nets -expr { [llength $obj(.loads)] > 5 }
  ```
- The following command returns all insts with `base_cell.name AND2`:
  ```
  get_db insts -expr {$obj(.base_cell.name) eq "AND2"}
  ```
- The following example presents the use of LEF with the `define_attribute` syntax:
  ```
  PROPERTYDEFINITIONS
  VIARULE myViaRule STRING ;
  END PROPERTYDEFINITIONS

  VIARULE via4Array
  LAYER metal4 ;
  ...
  LAYER metal5 ;
  ...
  PROPERTY myViaRule Test ;
  END via4Array
  ```

  `define_attribute` will be defined as:
  ```
  define_attribute myViaRule -category lef_property -data_type string -default Test -obj_type via_def_rule -skip_in_db -help_string {This was created by a LEF PROPERTYDEFINITION statement with the name "myViaRule". Any change to the value is only valid during the current session. The next read_db will recreate these attribute names and values from the LEF file only.}
  ```

  The following command will query the property:
  ```
  get_db via_def_rule:via4Array .myViaRule
  Test
  ```

Related Topics

- For details on LEF property definitions, see the 'Property Definitions' section on the LEF Syntax chapter in the *LEF/DEF Language Reference*.
- For information on how to set DB attributes, see the `set_db` command.
- For information on how to reset attribute values, see the `reset_db` command.
- For a list of objects attributes that can be queried for information or changed, see Common UI Database Object Information.

get_db Category Attributes

The root attributes in the `get_db` category are:

- get_db_display_limit

To set the value of any root attribute, use the `set_db` command:

```
set_db attribute_name value
```

To get the current value of any root attribute, use the `get_db` command:

```
get_db attribute_name
```

To get the current value of all attributes in the `get_db` category, use the following command:

```
get_db -category get_db
```

get_db_display_limit

`get_db_display_limit {integer}`

Default: `10`

Data_type: int, read/write

Controls the length of lists and number of objects displayed on the screen using queries that contain wildcards.

Note: This variable does not change the return value of `get_db` command.

Example

The following example sets the number of objects and length of the lists to 1 and then calls `get_db` to display instance attributes. `get_db` command outputs one instance object and every attribute list is truncated to one element.

```
set_db get_db_display_limit 1
get_db insts .*
```

```
Object: inst:TOP/I0
arcs:                   NC
area:                   0.7056
base_cell:              base_cell:BUFFD0BWP
bbox:                   {0.0 0.0 0.0 0.0}
.....
pg_base_pins:           base_pin:BUFFD0BWP/VDD ... (total length 2) #truncated as length of the list is set to 1
pg_pin_nets:            net:TOP/VDD ... (total length 2) #truncated as length of the list is set to 1
pins:                   pin:I0/Z ... (total length 2) #truncated as length of the list is set to 1
...
route_halo_top_layer:   {}
test_reset_inst:        1
```

Note: Number of objects exceeds limit. Use `set_db get_db_display_limit <value>` to control the number of objects evaluated by `get_db <object>.*` query.

is_attribute

```
is_attribute
[-help]
attribute_name
-obj_type obj_type
```

The `is_attribute` returns `true` if attribute `attribute_name` is defined for objects of type `obj_type`, `false` otherwise.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `is_attribute` parameter. For a detailed description of the command and all its parameters, use the man command: `man is_attribute`
`attribute_name`	Specifies the name of the attribute to be checked. *Data_type*: string, required
`-obj_type` *`obj_type`*	Specifies the object type of the attribute to be checked. *Data_type*: string, required

Examples

- The following `is_attribute` command returns `true` because the attribute `test_delete` is defined for `inst` type objects:

```
define_attribute test_delete -obj_type inst -data_type int -category test -help_string "test delete function" -default 1
is_attribute -obj_type inst test_delete
1
```

- The following commands first delete the attribute `test_delete` if it exists, then checks to see if the attribute exists (which it doesn't) and the 2nd `is_attribute` returns `false`:

```
if { [ is_attribute test_delete -obj_type inst ] } { delete_obj attribute:inst/test_delete }
is_attribute test_delete -obj_type inst
0
```

Related Topics

- How to query DB attributes, see the get_db command.
- For a list of object attributes that can be queried for information or changed, see the Common UI Database Object Information document.
- How to create user-defined attributes: define_attribute

reset_db

```
reset_db
[-help]
{ root_attribute_name | start [chain] | -category category_name }
[-quiet]
[-verbose]
```

Resets attribute value to a pre-defined default value. It returns a single value when resetting a single attribute, a list of values when resetting same attribute on different object instances, or a human readable table for multiple attributes. It also returns the values and number of attributes that were reset. The output is returned in the following format:

```
Attribute_name value
```

For example:

```
Place_xxx true
Add_filler  true
```

Note: The `-category` parameter of the `reset_db` command can be used to reset root attributes only. Both user-defined and system root attributes can be reset using the `-category` option.

Note: The attribute value can be reset if the attributes `default_value` is specified.

When the `-category` parameter is used, the command resets the values of both the hidden and unmapped attributes. In such a case, the command does not return the count of the reset attributes. The output in such cases will be like the following:

```
Some hidden attributes were also reset.
```

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `reset_db` parameter. For a detailed description of the command and all its parameters, use the man command: `man reset_db`
`root_attribute_name`	Resets the named root attribute. Wild-cards work. Attribute db-objects do not work. Attribute db-object names do not work. A tcl-list of root attribute names does not work.
`start`	Can be any object list (or collection) or *shorthand*. For example nets, insts... Attributes found by following *chain* off of this object list are reset. *Data_type*: tcl_obj, optional
`chain`	Specifies the attribute *chain* in the form *`.attribute_name1[.attribute_name2]...`* *Data_type*: string, optional

`-category` *`category_name`*	Specifies the root attribute category to be reset based. For example, the following command resets all the root attributes (including hidden and unmapped ones) to the default value in the `ccopt` category:
	`reset_db -category ccopt`
	One can find all the categories by:
	`get_db categories`
	`category_name` must be a single category name, no list of names, no wild-cards.
	Note: This applies only to the root level attributes and is ignored for the lower levels.
	Data_type: string, optional

`-quiet`	By default `set_db` is quiet. If the `set_db_verbose` root attribute is set to true, then the name of each object that is modified is echoed to the terminal (see the `-verbose` parameter's description for an example). However, `-quiet` forces `set_db` to be quiet even if `set_db_verbose` is true. *Data_type*: bool, optional

`-verbose`	Turns on verbose output that echoes the name of each object that is modified. This can be useful debug tool to ensure that the correct objects were affected. For example:
	`reset_db -verbose [get_db hinsts D*] .dont_touch`
	`Setting attribute of hinst 'DATA_BUS_MACH_INST': 'dont_touch' = false`
	`Setting attribute of hinst 'DECODE_INST': 'dont_touch' = false`
	`2 false`
	Note: The `set_db_verbose` root attribute can also be set to true, to force verbose output for all `set_db` and `reset_db` usage without requiring `-verbose`.
	Data_type: bool, optional

Examples

- The following resets the value of a user-defined instance attribute:

```
define_attribute test_reset_inst -obj_type inst -data_type int -category reset -help_string "test reset function" -default 1
1
set_db insts .test_reset_inst 10
5 true       #Five instances in the design
reset_db insts .test_reset_inst
5 1            #All 5 attributes are reset to default values
```

- The following command resets the root attributes using the `-category` parameter:

```
define_attribute test_reset1 -obj_type root -data_type int -category reset -help_string "test reset function" -default 1
define_attribute test_reset2 -obj_type root -data_type int -category reset -help_string "test reset function" -default 1
define_attribute test_reset3 -obj_type root -data_type int -category reset -help_string "test reset function" -default 1
set_db test_reset1 10
10
set_db test_reset2 10
10
set_db test_reset3 10
10
```

  On specifying:

```
reset_db -category reset
```

  The attributes are reset to their default values and a human readable table with new values is displayed.

```
test_reset1 1
test_reset2 1
test_reset3 1
```

- The following command resets all root attributes: `reset_db *`
- The following command resets the *place_status* attribute of all non-physical insts with `place_status` *fixed*: `reset_db [ get_db insts -if { ( .place_status == "fixed" ) && ! .is_physical } ] .place_status`

Related Topics

- For information on how to query DB attributes, see the `get_db` command.
- For information on how to set DB attributes, see the `set_db` command.
- For a list of objects attributes that can be queried for information or changed, see the Common UI Database Object Information document.
- For information on `set_db_verbose` root attribute: system Category Attributes

set_db

```
set_db
[-help]
start
[chain]
value
[-dbu]
[-index {index_type1 name1 [index_type2 name2 …]}]
[-quiet]
[-verbose]
```

Changes the specified attribute value for a database object. The `set_db` command takes a dual-ported object, dual-ported object list, or a collection as input, and then uses a period (.) to traverse to other related objects or to access attributes. You can use additional periods to traverse the data base schema.

Note: A Dual-Ported Object (DPO) is a `Tcl_Obj` in the C++ programming interface. The pointer is kept as a C++ pointer inside Tcl unless Tcl forces it to be evaluated as a string. In normal usage it never gets converted to a string which is more efficient, but if you do a `puts $var` or return the value to the shell, it will be converted to its dual string form.

- The string form is normally a useful name preceded by the `obj_type`, so a `base_cell` object string form might look like `base_cell:and2`.
- A `netlist` name will have the current design name included, so an instance named `i1/i2` in design top, would look like `inst:top/i1/i2`.
- An object with no name like a wire, will just show the pointer hex-value like `wire:0x22222`.

The `set_db` command does not create or delete objects.

For information on what object attributes are settable, see the Common UI Database Object Information document. It contains a description of all database objects and attributes.

The root object has the current design data under the `current_design` attribute, while technology and library data are normally directly accessible as root attributes (For example, layers and libraries). The root object is the default starting point if there is no input list or collection given.

You can set any root attribute in any of the following ways:

- `set_db / .flow_mail_to name`
  where, '/' is the notation for the root object , or,
- `set_db flow_mail_to name`
  This implicitly starts at the root.

For example, you can use the following command to change the design text font:
`set_db current_design .texts.font_number  4`

Note: The `current_design` is implicitly created by the `init_design` or the `read_db` commands.

Frequently used `current_design` attributes like `insts` or `hinsts` (hierarchical insts) have root attribute aliases (or shortcuts). These attributes can be accessed by just typing the root attribute alias name.
For example, the following two commands set all insts .place_status to fixed.

```
set_db current_design .insts.place_status fixed
set_db insts .place_status fixed
```

Some of the more commonly used aliases from the current_design attributes include: insts, hinsts, nets, hnets, ports, hpins, pins.

The `set_db` command supports auto-complete when you press the <`Tab`> key. It will give a list of all matching attributes starting with the letters entered so far, or it will complete the name if it is unique.

For example:
`set_db plan_design_use<Tab>`
returns `plan_design_use_guide_boundary plan_design_use_sdp_group`

This command does not support the usage of embedded Tcl command as operands. Use the `-expr` parameter of the `get_db` command for this requirement. For information on how to access DB objects, see the `get_db` command.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `set_db` parameter.
start	The starting objects or attributes. Can be an object list, collection, root (/), or root attribute. *Data_type*: tcl_obj, required
chain	Specifies a *chain* in the form `[.attribute1[.attribute2]...]`. The attribute names must be valid for each level of the traversal in the *chain*, where the first name is valid for the start object. For example, `set_db $my_pins .inst.place_status fixed` where `.inst` is an attribute of an pin object and `.place_status` is an attribute of an inst object. *Data_type*: string, optional
value	Specifies new value for the specified object attribute elements. The value type depends on which object attribute type is being modified. *Data_type*: string, required
`-dbu`	Indicates that attribute values (coord, pt, rect, polygon, area, ...) should be in database units. *Default*: The default is microns (or microns**2). *Data_type*: bool, optional
`-index {`*index_type1 name1* `[`*index_type2 name2* `…]}`	
	Some attributes represent an array of values. For example, a pins .slack_max_rise attribute has a different value for each analysis_view. With no index, all the values defined for that attribute are modified by the `set_db` command. If you want to change a specific value, you must include the index value. The index values are pairs of obj_type names (for example, analysis_view, clock, skew_group) followed by the name. The word "view" is accepted as an abbreviation for "analysis_view". For example, some ccopt attributes can take both a view and clock index like this: `-index "view func_wc_cworst clock clk1"` *Data_type*: tcl_obj, optional
`-quiet`	By default `set_db` is quiet. If the `set_db_verbose` root attribute is set to true, then the name of each object that is modified is echoed to the terminal (see the `-verbose` parameter's description for an example). `-quiet` forces `set_db` to be quiet even if `set_db_verbose` is true. This is intended for internal scripts that want to be certain that the verbose output is off. *Data_type*: bool, optional

`-verbose`	Turns on verbose output that echoes the name of each object that is modified. This can be useful during interactive debugging to ensure that the correct objects were affected. For example:
	`set_db -verbose [get_db hinsts D*] .dont_touch false`
	`Setting attribute of hinst 'DATA_BUS_MACH_INST': 'dont_touch' = false`
	`Setting attribute of hinst 'DECODE_INST': 'dont_touch' = false`
	`2 false`
	Note: The `set_db_verbose` root attribute can also be set to true, to force verbose output for all `set_db` and `reset_db` usage without requiring `-verbose`.
	Data_type: bool, optional

Examples

- The following command sets a value of root attribute.
  ```
  set_db place_global_solver_effort high
  ```
- The following commands define an instance attribute and change its value to 0.
  ```
  define_attribute myattr -default 1 -obj_type inst -data_type int -help_string "my attribute" -category my_cat
  set_db insts .myattr 0
  ```
- The following command sets an instance attribute value to 5 only if its current value is 0.
  ```
  set_db [get_db insts -if {.myattr == 0}] .myattr 5
  ```

Related Topics

- For information on how to query DB attributes, see the `get_db` command.
- For information on how to reset attribute values, see the `reset_db` command.
- For a list of objects attributes that can be queried for information or changed, see the Common UI Database Object Information document.

sort_db

```
sort_db
[-help]
obj_list
[attribute_list]
[-descending]
[-dictionary]
```

Returns a sorted DPO (dual-ported object) list based on the contents of one or more attributes. It is based on `lsort` using get_db to access various attributes.

This command sorts objects only based on the attributes that have a "simple" data type of string, float, integer, or object. If it is an object, it is implicitly sorted by the `.name` field and treated as a string. Numeric values are sorted numerically, and string values are sorted using the ASCII values like `lsort`.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `sort_db` parameter. For a detailed description of the command and all its parameters, use the man command: `man sort_db`

`obj_list`	A DPO list to be sorted. If the first element of the specified list is not a DPO, then it is treated as a root attribute name and passed directly to `get_db` to return a DPO list. So, the following two commands have identical results:
	`sort_db insts`
	`sort_db [get_db insts]`
	Data_type: tcl_obj, required

`attribute_list`	Sorts the DPO list by the specified attribute values. The first attribute in the *`attribute_list`* has the highest sort precedence. Each attribute can be a chain identical to what `get_db` supports. A leading "." is optional, so all the following commands presenting the use of `attribute_list` specify legal arguments for an inst list:
	`sort_db insts name`
	`sort_db insts .name`
	`sort_db insts {location.x location.y}`
	`sort_db insts {.location.x .location.y}`
	Data_type: string, optional

`-descending`	Sorts the list of objects by the descending order. *Data_type*: bool, optional
`-dictionary`	Sorts string attribute values by the dictionary order (the same as `lsort -dictionary`). The casing of characters is ignored except as a tie-breaker. If two strings contain embedded numbers, the numbers are compared as integers, not characters. For example, `bigBoy` is sorted between `bigbang` and `bigboy`, and `x10y` is sorted between `x9y` and `x11y`. *Data_type*: bool, optional

Examples

- The following command sorts all the insts by name:
  `sort_db insts`
- The following command sorts the pins based on their slack. The pins with the same slack are further sorted by the x-location:
  `sort_db [get_db pins] {.slack .location.x}`

Related Topics

- get_db
- set_db

write_preserves

```
write_preserves
[-help]
[-dont_touch]
[-dont_use]
[-obj_types {design hinst module inst net hnet pin hpin base_cell}]
[-out_file string]
```

Writes preserve constraints in a consumable TCL file so that the exported file can sourced.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `write_preserves` parameter. For a detailed description of the command and all its parameters, use the man command: `man write_preserves`
`-dont_touch`	Writes the preserve constraints only for the non-default attributes related to the `dont_touch` object. *Data_type*: bool, optional
`-dont_use`	Writes the preserve constraints only for the non-default attributes related to the `dont_use` object. *Data_type*: bool, optional
`-obj_types {design hinst module inst net hnet pin hpin base_cell}`	
	Writes the preserve constraints only for the specific object types. *Data_type*: (enum)+, optional
`-out_file string`	Displays the TCL file name. *Data_type*: string, optional

9

Delay Calculation Commands and Attributes

- add_decap_cell_candidates
- add_decaps
- delaycal Category Attributes
- delete_decaps
- delete_exclude_net_attribute
- report_decap_cell_candidates
- report_decaps
- report_delay_calculation
- reset_decap_cell_candidates
- write_exclude_nets
- write_set_load

add_decap_cell_candidates

```
add_decap_cell_candidates
[-help]
{ cell_name capacitance [ cell_name capacitance ... ] | -in_file fileName }
```

Defines the cells that can be used by the `add_decaps` command to insert decoupling capacitance.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `add_decap_cell_candidates` parameter. For a detailed description of the command and all of its parameters, use the man command: man `add_decap_cell_candidates`.
`cell_name capacitance [ cell_name capacitance ... ]`	Specifies the base cell name(s) and capacitance value(s) in femtofarads (fF) for the cell candidate(s).
`-in_file` *`fileName`*	Defines the cells and capacitance values contained in the specified file as decoupling capacitance cell candidates. File format is: one or more lines of `cell_name capacitance`

Example

The following commands define two decoupling capacitance cell candidates: `DECAP1` has a capacitance value of 10fF, and `DECAP8` has a capacitance value of 5fF.

```
add_decap_cell_candidates DECAP1 10
add_decap_cell_candidates DECAP8 5
```

In addition, the following commands do the same thing:

```
add_decap_cell_candidates DECAP1 10 DECAP8 5
add_decap_cell_candidates "DECAP1 10 DECAP8 5"
add_decap_cell_candidates {DECAP1 10 DECAP8 5}
add_decap_cell_candidates [ list DECAP1 10 DECAP8 5 ]
```

Related Topics

- Placing the Design chapter in the *Innovus System User Guide*
  - Adding Decoupling Capacitance
- `report_decap_cell_candidates`
- `reset_decap_cell_candidates`

add_decaps

```
add_decaps
[-help]
-total_cap total_cap_in_fF
[-cells cellName ]
[-place_status]
[-area llx lly urx ury |  -exclude_areas {{ llx lly urx ury } ...}]
[-effort [low | high]]
[-prefix prefixName ]
[-ignore_drc]
[-log logName]
[-max_iteration number]
[-in_file fileName]
[-pg_net netName]
```

Adds decoupling capacitance instances to the design. The software chooses from decoupling capacitance cell candidates and adds instances until their combined total capacitance value equals the user-specified value.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `add_decaps` parameter. For a detailed description of the command and all of its parameters, use the man command: `man add_decaps`.
`-place_status`	Added decap instances are marked as `place_status fixed` (otherwise: `place_status placed`)
`-area llx lly urx ury`	Defines an area within the design in which to add the decoupling capacitance. *Default:* Entire design area
`-cells cellName …`	The base cell names to be used for decoupling capacitance insertion. The cells must first be defined using the `add_decaps_cell_candidates` command. *Default:* All cells defined with previous `add_decap_cell_candidates` commands

`-effort [low | high]`	Specifies the approach to use when inserting decoupling capacitance. **low:** Uses a homogeneous approach to add the decoupling capacitance. That is, the software treats all areas the same when adding decoupling capacitance cells. **high:** Analyzes the peak current of the instances and tries to add more cells to the areas with high peak current density. *Default:* `low`
`-exclude_areas { llx lly urx ury } …`	Defines a blockage area within which decoupling capacitance cannot be inserted. You can specify one or more blockage areas. **Note:** The parameters `-exclude_areas` and `-area` are mutually exclusive. When both parameters are specified on the command line, the tool throws an error.
`-ignore_drc`	Disables correction of DRC errors.
`-in_file fileName`	Performs decoupling capacitance insertion using the information in the specified log file (created from a previous `add_decaps` session). This file contains information for adding decoupling capacitance cells and for deleting existing decoupling capacitance cells.

`-log` *logName*	Outputs a log file with the specified name that contains information on where the decoupling capacitance cells were added. It can also contain information on where cells were deleted due to DRC violations.
	The log file can be used for tracking the location of decoupling capacitance and for debugging purposes. It can also be used to insert decoupling capacitance cells in a different session.
	The log file is output in the following format:
	`ADD` *cellName locationX locationY* `[` *orientation* `] [` *instanceName* `]`
	`DEL` *instanceName*
	It contains the cell name and the location where an instance of it was placed. It can also contain the orientation of the instance, and a name for the instance, if that name has not been used already.
	For example, the following line shows that a new instance of `CELL_8` with the name `Instance100` was added at the location 100 200, with an orientation of R0:
	`ADD CELL_8 100 200 R0 Instance100`

`-max_iteration` *number*	Specifies the maximum number of iterations of DRC fixing. *Default*: 10
`-prefix` *prefixName*	Defines the prefix to use for the decoupling capacitance cells. *Default:* `DECAP`
`-pg_net` *netName*	Adds decap cells on the specified power rail. *Default:* Add decap cells irrespective of power nets.
`-total_cap` *total_cap_in_fF*	Specifies the total decoupling capacitance to add to the design, in femtofarads (fF).

Command Order

Use this command after defining decoupling capacitance cell candidates with the `add_decap_cell_candidates` command.

Examples

- The following commands add 1000 fF of capacitance to the design using `DECAP1` and `DECAP8` cells, in that order:

```
add_decaps -total_cap 1000 -cells DECAP1 DECAP8
add_decaps -total_cap 1000 -cells "DECAP1 DECAP8"
add_decaps -total_cap 1000 -cells {DECAP1 DECAP8}
add_decaps -total_cap 1000 -cells [ list DECAP1 DECAP8 ]
```

- The following command adds 6367.341 fF of capacitance to the specified area within the design, and marks the placement status of all instances as fixed:

```
add_decaps -total_cap 6367.341 -area 600 600 900 900 -place_status
```

Related Information

- Placing the Design chapter in the *Innovus System User Guide*
  - Adding Decoupling Capacitance
- `delete_decaps`
- `report_decaps`

delaycal Category Attributes

The root attributes in the "delaycal" category are:

- delaycal_accuracy_level
- delaycal_advanced_pin_cap_mode
- delaycal_advanced_node_pin_cap_settings
- delaycal_combine_mmmc
- delaycal_degrade_slew_on_early_nets
- delaycal_default_net_delay
- delaycal_default_net_load
- delaycal_enable_high_fanout
- delaycal_enable_quiet_receivers_for_hold
- delaycal_enable_si
- delaycal_enable_wire_load_model
- delaycal_equivalent_waveform_model
- delaycal_equivalent_waveform_type
- delaycal_equivalent_waveform_model_for_timing_check
- delaycal_honor_slew_propagate_constraint
- delaycal_ignore_net_load
- delaycal_input_transition_delay
- delaycal_irdrop_data_type
- delaycal_early_irdrop_data_type
- delaycal_late_irdrop_data_type
- delaycal_library_interpolation_mode
- delaycal_irdrop_window_based
- delaycal_report_out_bound
- delaycal_signoff_alignment_settings

- delaycal_slew_out_bound_limit_high
- delaycal_slew_out_bound_limit_low
- delaycal_skip_slew_merge_from_disabled_path
- delaycal_socv_accuracy_mode
- delaycal_socv_lvf_mode
- delaycal_socv_machine_learning_level
- delaycal_socv_use_lvf_tables
- delaycal_support_output_pin_cap
- delaycal_support_wire_load_model
- delaycal_timing_create_clock_use_ideal_slew
- delaycal_use_default_delay_limit

To set the value of any root attribute, use the `set_db` command:

```
set_db attribute_name value
```

To get the current value of any root attribute, use the `get_db` command:

```
get_db attribute_name
```

To get the current value of all attributes in the `power` category, use the following command:

```
get_db -category delaycal
```

For delay calculation reports, use the following command:

- `report_delay_calculation`

delaycal_accuracy_level

`delaycal_accuracy_level` *level*

Default: 0, *Min*: 0, *Max*: 4

Sets the accuracy level for delay calculations.

You can use this attribute to enable new enhancements for different process nodes, without using the `design_process_node` parameter.

delaycal_advanced_pin_cap_mode

```
delaycal_advanced_pin_cap_mode {0 | 1 | 2}
```

Default: 0

Simulates the current stage and receiver stages together, such that the back miller effect is modeled dynamically.

When set to 1, the receivers are no longer modeled as simple non-linear current source models. This results in increasing the delay calculation accuracy significantly as the modeling back miller effect improves.

When set to 0, the receiver current sources are computed using ECSM/CCS capacitance tables with the estimated input slew.

delaycal_advanced_node_pin_cap_settings

```
delaycal_advanced_node_pin_cap_settings {true | false}
```

Default: `false`

Enables advanced node pin cap settings.

delaycal_combine_mmmc

```
delaycal_combine_mmmc {none | early_late | early_late_corner}
```

Default: `early_late_corner`

Specifes the vdc mode to be used.

Specifies whether delay calculation runs are to be combined for delay calculation simulations. This attribute provides user control for improving the runtime of AAE for all analysis types by combining runs for early and late simulations into a single simulation. The views with common delay corners may also be combined into a single delay calculation simulation for added runtime improvement.
When none is specified, the software does not combine any delay calculation runs. Individual delay calculation simulations are performed for early and late for example, which will result in significant CPU performance degradation due to the increased number of simulation runs performed.
When `early_late` is specified, the software combines early and late simulations of a single corner into one delay calculation simulation and will improve CPU performance by avoiding unnecessary delay calculation simulations.
When early_late_corner is specified, the software combines early / late simulations of a single corner, like with the `early_late` setting,

but also combines views with common delay corners into one single delay calculation simulation. For example, the software will combine views having different `set_case_analysis` statements in one single simulation. This can result in significant improvement in CPU performance by avoiding unnecessary delay calculation simulations.

delaycal_degrade_slew_on_early_nets

```
delaycal_degrade_slew_on_early_nets {true | false}
```

Default: `true`

Controls whether slews in early path delay calculation are degraded as they pass across the interconnect from driver to receiver. You can specify a value of `false` to turn off interconnect slew degradation for early paths and have the driver slews copied to the receivers. Early path delay calculation affects all Hold paths in best-case/worst-case analysis mode, capture clock paths in on-chip variation Setup analysis, and clock launch and data paths in on-chip variation Hold analysis.

delaycal_default_net_delay

```
delaycal_default_net_delay delay_value
```

Type: `String`
Default: 1000ps

Sets the default net delay for delay calculation. The software uses this default value for nets that exceed 1,000 terminals. Delay calculation is performed for nets with fewer than 1,000 terminals.

delaycal_default_net_load

```
delaycal_default_net_load load_value
```

Type: `String`

Default: 0.5pF

Sets the default net load for delay calculation. You can use this attribute to specify the default net load value that is used for nets that exceed 1,000 terminals.

delaycal_enable_high_fanout

```
delaycal_enable_high_fanout {true | false}
```

Default: `false`

When this attribute is set to `true`, the default net delay will be annotated on high fanout nets. A net will be treated as a high fanout net when the fanout is more than the limit specified using the `delaycal_use_default_delay_limit` attribute. The default value of this attribute is 1000 fanouts. The default net delay is also specified using this attribute. The default net delay is 1ns. The cell delay will use the lumped capacitance that is specified using the `delaycal_default_net_load` attribute for table lookup. The default net load is 0.5pF. The driver slew will use the default value of 0ps , which can be changed by using the `delaycal_input_transition_delay` attributes and there will be no slew degradation along the interconnect.

delaycal_enable_quiet_receivers_for_hold

```
delaycal_enable_quiet_receivers_for_hold {true | false}
```

Default: `false`

Forces some receivers to be static (“quiet”).

When set to `true`, the software assumes that the receivers are not switching - to compute the worst-case delays for early analysis. This yields more conservative early delays.

When set to `false`, the software assumes that the receivers may switch.

delaycal_enable_si

```
delaycal_enable_si {true | false}
```

Default: `false`

Enables SIAware delay calculation that also includes cross-talk induced delays.

Note: SIAware delay calculation requires the design to be in Multi-Mode, Multi-Corner (MMMC) Mode and the analysis type to be set to on-chip variations.

delaycal_enable_wire_load_model

```
delaycal_enable_wire_load_model {0 | 1}
```

Type: `Integer`

Default: 0

Enables support of Liberty wire-load models and related SDC commands.

delaycal_equivalent_waveform_model

```
delaycal_equivalent_waveform_model {none | no_propagation | propagation}
```

Default: `none`

Specifies the equivalent waveform model to be used.

delaycal_equivalent_waveform_type

```
delaycal_equivalent_waveform_type {moments | simulation}
```

Default: `moments`

Specifies the equivalent waveform model (EWM) to be used:

The options are described below:

- `moments`: Enables moment-based equivalent waveform model.
- `simulation`: Enables simulation-based equivalent waveform model. The simulation-based EWM is required for predicting delay in noisy waveforms.

It is recommended to use simulation-based EWM as it is more accurate than moment-based EWM.

delaycal_equivalent_waveform_model_for_timing_check

```
delaycal_equivalent_waveform_model_for_timing_check {true | false}
```

Default: `false`

Models the waveform distortion effect for timing check value look ups at both the data and clock pins.

The model uses simulations to convert the distorted waveforms into equivalent waveforms (pre-driver waveforms in the libraries) by matching the pin CCB output waveforms. This feature requires the CCSN library.

delaycal_honor_slew_propagate_constraint

```
delaycal_honor_slew_propagate_constraint {true | false}
```

Default: `true`

Determines whether to propagate the slew from the disabled timing arcs to the output pin.

When this attribute is set to `false`, the slew from the disabled timing arc affects the delay of the next stage. Setting this attribute to false results in more accurate slew computation for the disabled timing arcs but can take longer run time.

When set to `true`, the timing constraints defined using commands such as `set_disable_timing` and `set_case_analysis` impact the slew propagation, and the slew from the disabled timing arcs is not propagated.

delaycal_ignore_net_load

```
delaycal_ignore_net_load {true | false}
```

Default: `false`

Ignores the net load for delay calculation.

When set to `true`, this attribute removes the default system requirement for having RC parasitics available for delay calculation.

This attribute is used to get initial design timing when the design is not in a proper state to generate extracted results and an external SPEF source of parasitics is not available. The results obtained by enabling this attribute are equivalent to those obtained using a zero-cap, zero-resistance wire-load model for estimating delays.

delaycal_input_transition_delay

```
delaycal_input_transition_delay " "
```

Type: String

Default: 4ps

Sets the default input transition delays and the ideal clock transition time for delay calculation.

The default transition will be used in those cases where the transition is not defined (using `set_input_transition`, `set_annotated_transition`, or any driving cell). For all the ports/pins that do not have any driving cell in fanin (or `set_input_transition`/`set_annotated_transition` settings) will also use this default slew value.

It is mandatory to specify time units for this command. Valid time units are ps, ns, etc.
In addition, the default input transition value is applied to those pins to which the slew was not propagated. For example:

- Tiehi or tielo pins
- Macro output pins, which do not have a related input pin

A macro output pin, which has a related input pin will most likely have slew propagation from the related input pin to the output pin.

- Macro output pins for which all the timing arcs are disabled
- Input ports without a user-specified transition value

delaycal_irdrop_data_type

```
delaycal_irdrop_data_type {best_average | worst | best | average | worst_average}
```

Default: `worst`

Allows you to set early/late IR drop EIV values for different EIV voltages.

delaycal_early_irdrop_data_type

```
delaycal_early_irdrop_data_type {best_average | worst | best | average | worst_average}
```

Default: `worst`

Allows you to set early IR drop EIV values for different EIV voltages.

delaycal_late_irdrop_data_type

```
delaycal_late_irdrop_data_type {best_average | worst | best | average | worst_average}
```

Default: `worst`

Allows you to set late IR drop EIV values for different EIV voltages.

delaycal_library_interpolation_mode

```
delaycal_library_interpolation_mode {linear | non_linear}
```

Default: `linear`

Controls library scaling mode - `linear` or `non_linear` - for the interpolation flow.

delaycal_irdrop_window_based

```
delaycal_irdrop_window_based {none | late | early | both}
```

Default: `both`

Searches window-based EIV values from the EIV database.

delaycal_report_out_bound

```
delaycal_report_out_bound {true | false}
```

Default: `false`

Generates a report that contains a list of index values (input transition) in the delay tables that are beyond the index range. The index values detected by this attribute can cause inaccurate delay values to be calculated. The report generated by this attribute can be used for debugging.

When this attribute is set to `true`, the extrapolation information will be dumped into the following two files:

- <design-name>.dc.outbound.slew.early.<view-name>.txt
- <design-name>.dc.outbound.slew.late.<view-name>.txt

The software writes a warning message in the log files noting that there are input transition values that are out of the index range of the timing table. The warning message also provides the file name for viewing the detail report.

delaycal_signoff_alignment_settings

```
delaycal_signoff_alignment_settings {true | false}
```

Default: `false`

Enables signoff alignment settings.

delaycal_slew_out_bound_limit_high

```
delaycal_slew_out_bound_limit_high double
```

Default: 3.40282e+38

Limits the maximum slew used in the analysis to the specified value times max characterized slew in a library.

delaycal_slew_out_bound_limit_low

`delaycal_slew_out_bound_limit_low` *`double`*

Default: 0.5

Limits the minimum slew used in the analysis to the specified value times min characterized slew in a library.

delaycal_skip_slew_merge_from_disabled_path

`delaycal_skip_slew_merge_from_disabled_path {true | false}`

Default: `false`

Avoids merging slews from unconstrained, segmented paths and black-box output pin.

When set to `true`, the software does not allow merging transitions from disabled paths.

delaycal_socv_accuracy_mode

`delaycal_socv_accuracy_mode {low | high}`

Default: `low`

Allows the ability to control levels of accuracy.

This attribute specifies level-based accuracy effort for delays and slews. You can use one of the following options:

- `low`: Ignores correlation between slews and delays.
- `high`: Considers skewness of non-gaussian distribution into account for computing quantiles. The skewness is local to one stage only.

delaycal_socv_lvf_mode

`delaycal_socv_lvf_mode` *`string`*

Default: `moments`

Controls SOCV LVF analysis modes. You can specify - `moments`, `early_late`, or `average` - options.

delaycal_socv_machine_learning_level

```
delaycal_socv_machine_learning_level {0 | 1}
```

Default: 0

Enables the SOCV machine learning mode.

For accurate SOCV analysis, the software creates several statistical models, which are not available with standard LVF library, such as correlation between cells delay and output slew, conversion of moments from late/early sigma tables, and effects of waveform distortion on sigma and mean shift. When this attribute is set to 1, the software enables several machine learning algorithms to generate such models internally.

delaycal_socv_use_lvf_tables

```
delaycal_socv_use_lvf_tables string
```

Default: `all`

Controls which type of variations – `all`, `delay`, `slew`, or `constraint` - are considered during analysis.

delaycal_support_output_pin_cap

```
delaycal_support_output_pin_cap {true | false}
```

Default: `true`

Includes the output pin capacitance when calculating delay values. When set to 0, the software excludes only the pin capacitance of the driver for which you are calculating the delay.

delaycal_support_wire_load_model

```
delaycal_support_wire_load_model {0 | 1}
```

Type: Integer

Default: 0

Enables support of Liberty wire-load models and related SDC commands.

delaycal_timing_create_clock_use_ideal_slew

```
delaycal_timing_create_clock_use_ideal_slew {true | false}
```

Default: `false`

Enables the use of ideal slew at the inputs of the clock driver cells.

When set to `true`, the software detects created clocks, and uses zero input slew. Other slew values, including those set using `set_annotated_transition` command, will be ignored.

When set to `false`, the software honors `set_annotated_transition` command settings.

delaycal_use_default_delay_limit

```
delaycal_use_default_delay_limit value
```

Type: Integer

Default: 1000

Sets the pin number threshold for nets for delay calculation. Nets that have more than the specified number of pins are excluded from delay calculation, and use the default delay values instead. The net default delay values are:

- Net delay: 1 ns
- Transition time: 0 ps
- Net load: 0.5 pF

Note: In post route stage, default delay calculation does not honor this attribute. Slews and delays are computed based on actual parasitics.
To change the default delay values, set the following attributes:

- `delaycal_default_net_delay`
- `delaycal_default_net_load`
- `delaycal_input_transition_delay`

delete_decaps

```
delete_decaps
[-help]
```

Deletes all decoupling capacitance cells. You can use this command to delete decoupling

capacitance cells that you added using the `add_decaps` command.

Parameters

`-help`	Outputs a brief description of the `delete_decaps` command. For a detailed description of the command, use the man command: `man delete_decaps`.

Examples

The following command deletes all of the decoupling capacitance cells in a design:

```
delete_decaps
```

Related Topics

- Placing the Design chapter in the *Innovus System User Guide*
  - Adding Decoupling Capacitance
- `add_decaps`
- `report_decaps`

delete_exclude_net_attribute

```
delete_exclude_net_attribute [-help]
```

Clears excluded net settings - any nets that were previously excluded from delay calculation are included. You do not have to rebuild the timing graph to include the previously-excluded nets.

Parameter

`-help`	Outputs a brief description of the `delete_exclude_net_attribute` command. For a detailed description of the command, use the man command: `man delete_exclude_net_attribute`.

Examples

The following command clears excluded nets such that previously excluded nets are now included again in delay calculation:

```
delete_exclude_net_attribute
```

Related Information

- `write_exclude_nets`

report_decap_cell_candidates

```
report_decap_cell_candidates
[-help]
[-out_file outputFileName]
```

Reports the available cells for decoupling capacitance insertion. You use this command to report decoupling capacitance candidates previously specified using the add_decap_cell_candidates command.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `report_decap_cell_candidates` parameter. For a detailed description of the command and all of its parameters, use the man command: `man report_decap_cell_candidates`.
`-out_file` *filename*	Outputs a list of the available cells for decoupling capacitance insertion to the specified file. This file can then be used by the `add_decap_cell_candidates` command to define cell candidates in a different session, or after cell candidates have been deleted.

Example

Following is a sample report generated by using the `report_decap_cell_candidates` command:

```
############################################################
# Generated by:
# OS:          Linux x86_64
# Generated on: Tue Jan 25
# Design:      dtmf_chip
# Command:      report_decap_cell_candidates
############################################################
#The available decoupling capacitance cells:
#--------------------------
#Cell Name    Cap (fF)
#--------------------------
```

```
DECAP1 10
DECAP8 5
#-------------------------
```

Related Topics

- Placing the Design chapter in the *Innovus System User Guide*
  - Adding Decoupling Capacitance
- `add_decap_cell_candidates`
- `reset_decap_cell_candidates`

report_decaps

```
report_decaps
[-help]
-area llx lly urx ury
```

Reports placed decoupling capacitance cells in the specified area.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `report_decaps` parameter. For a detailed description of the command and all of its parameters, use the man command: `man report_decaps`.
`-area` *`llx`* *`lly`* *`urx`* *`ury`*	Defines the area for which to report placed decoupling capacitance cells. *lly* Specifies the area's lower left y coordinate. *urx* Specifies the area's upper right x coordinate. *ury* Specifies the area's upper right y coordinate.

Example

The following command reports placed decoupling capacitance cells in the specified area:

```
report_decaps -area 250 250 1000 1000
```

Related Topics

- Placing the Design chapter in the *Innovus System User Guide*
  - Adding Decoupling Capacitance
- `add_decaps`
- `delete_decaps`

report_delay_calculation

```
report_delay_calculation
[-help]
[-active_arcs]
-from port_or_pin_name
[-max]
[-min]
[-out_file string]
[-si]
[-show_all_attackers]
[-thresholds]
-to port_or_pin_name
[-view viewname]
[-voltage]
[-waveform]
```

Reports the delay calculation information for a cell or net timing arc. You can use this command to report delay calculation details for the specified timing arc. The delay calculation report may be useful for debugging timing or for verifying timing data.

You can run the `report_timing` command before running `report_delay_calculation` to have the correct timing window and aggressor slew.

Note: The `report_delay_calculation` command does not support AOCV derating.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `report_delay_calculation` parameter. For a detailed description of the command and all of its parameters, use the man command: `man report_delay_calculation`.
`-active_arcs`	Disables reporting of all the non-active parallel delay arcs and reports only the valid check arcs.

`-from` *port_or_pin_name*	Specifies the name of the From port or pin. If the From port or pin is output of a cell and To port or pin is input of a cell, delay is calculated for the net and the delay type in the report is 'net delay.' If the From port or pin is input of a cell and To port or pin is output of a cell, delay is calculated for the cell and the delay type in the report is 'cell delay.'
`-max`	Indicates that the delay calculation report is for the worst-case operating conditions in min-max mode. If neither `min` nor `max` arguments are specified, then the default mode is set to `max`.
`-min`	Indicates that the delay calculation report is for the best-case operating conditions in min-max mode. If neither `min` nor `max` arguments are specified, then the default mode is set to `max`.
`-out_file` *string*	Specifies the name of the output file in which to save the delay calculation report.
`-si`	Reports signal integrity (SI) related information.
`-show_all_attackers`	Includes all the attackers in the report.
`-thresholds`	Displays the rise delay, rise slew, fall delay, fall slew, and slew delay threshold values for the From and To pins.
`-to` *port_or_pin_name*	Specifies the name of the To port or pin. If the From port or pin is output of a cell and To port or pin is input of a cell, delay is calculated for the net and the delay type in the report is 'net delay.' If the From port or pin is input of a cell and To port or pin is output of a cell, delay is calculated for the cell and the delay type in the report is 'cell delay.'
`-view` *viewname*	Specifies the view name for which to print the delay calculation report when using the multi-mode multi-corner (MMMC) mode.
`-voltage`	Displays the voltage of the cell or net timing arc.
`-waveform`	Reports the waveforms of the pins.

Example

- The following command shows the cell delay calculation report for the From pin I1/A and To pin I1/Y:

```
report_delay_calculation -from I1/A -to I1/Y
From pin : I1/A
To Pin : I1/Y
Cell : S2INVV1D1
Library : test
Arc sense : negative unate
Delay type : cell delay
---------------------------------------
RC Summary for net N2
---------------------------------------
Number of capacitance : 7
Net capacitance : 0.026057 pF
Total rise capacitance: 0.027172 pF
Total fall capacitance: 0.027169 pF
Number of resistance : 7
Total resistance : 1094.135127 Ohm
---------------------------------------
Rise Fall
---------------------------------------
Input transition time : 0.118700 ns 0.151800 ns
Effective capacitance : 0.026337 pF 0.025975 pF
Cell delay : 0.325700 ns 0.280800 ns
Output transition time : 0.298600 ns 0.204500 ns
---------------------------------------
```

- The following command shows the net delay calculation report for the from pin ff1/Q and to pin A1/buf1/A with waveforms:

```
tempus> report_delay_calculation -from ff1/Q -to A1/buf1/A -waveform


From pin : ff1/Q
Cell : DFFX2
Library : gpdk045bc
To pin : A1/buf1/A
Cell : BUFX2
Library : gpdk045bc
Delay type: net
```

```
------------------------------------------------------------
From pin waveforms
------------------------------------------------------------
Rise Waveform : { 0.00e+00 0.00000 5.35e-10 0.05400 1.07e-09 0.10800 1.61e-09
0.16200
2.14e-090.21600 2.68e-09 0.27000 3.21e-09 0.32400 3.75e-09 0.37800 4.28e-09
0.43200
4.82e-09 0.486005.35e-09 0.54000 5.89e-09 0.59400 6.42e-09 0.64800 6.96e-09
0.70200
7.50e-09 0.756008.03e-09 0.81000 8.57e-09 0.86400 9.10e-09 0.91800 9.64e-09
0.97200
1.02e-08 1.02600 1.07e-08 1.08000 }

Fall Waveform : { 0.00e+00 1.08000 3.69e-10 1.02600 7.39e-10 0.97200 1.11e-09
0.91800
1.48e-09 0.86400 1.85e-09 0.81000 2.22e-09 0.75600 2.59e-09 0.70200 2.96e-09
0.64800
3.32e-09 0.59400 3.69e-09 0.54000 4.06e-09 0.48600 4.43e-09 0.43200 4.80e-09
0.37800
5.17e-09 0.32400 5.54e-09 0.27000 5.91e-09 0.21600 6.28e-09 0.16200 6.65e-09
0.10800
7.02e-09 0.05400 7.39e-09 -0.00000 }
------------------------------------------------------------
To pin waveforms
------------------------------------------------------------
Rise Waveform : { 2.73e-14 0.00000 5.35e-10 0.05318 1.07e-09 0.10793 1.61e-09
0.16125
2.14e-09 0.21587 2.68e-09 0.26930 3.21e-09 0.32383 3.75e-09 0.37734 4.28e-09
0.43179
4.82e-09 0.48538 5.35e-09 0.53975 5.89e-09 0.59341 6.42e-09 0.64772 6.96e-09
0.70144
7.50e-09 0.75570 8.03e-09 0.80946 8.57e-09 0.86368 9.10e-09 0.91748 9.64e-09
0.97166
```

```
1.02e-08 1.02549 1.07e-08 1.07965 }

Fall Waveform : { 2.38e-14 1.08000 3.69e-10 1.02719 7.39e-10 0.97209 1.11e-09
0.91911
1.48e-09 0.86416 1.85e-09 0.81104 2.22e-09 0.75623 2.59e-09 0.70298 2.96e-09
0.64828
3.32e-09 0.59493 3.69e-09 0.54033 4.06e-09 0.48688 4.43e-09 0.43237 4.80e-09
0.37885
5.17e-09 0.32440 5.54e-09 0.27081 5.91e-09 0.21643 6.28e-09 0.16279 6.65e-09
0.10846
7.02e-09 0.05476 7.39e-09 0.00048 }
-------------------------------------------------------------


-------------------------------------------------------------
RC Summary for net n2
-------------------------------------------------------------
Number of capacitance : 19
Net capacitance : 0.714975 pF
Total rise capacitance: 0.716177 pF
Total fall capacitance: 0.716179 pF
Number of resistance : 24
Total resistance : 101.624695 Ohm
-------------------------------------------------------------
Rise Fall
-------------------------------------------------------------
Net delay : 0.002500 ns 0.002300 ns
From pin transition time: 8.565800 ns 5.910500 ns
To pin transition time : 8.568400 ns 5.913000 ns
-------------------------------------------------------------
```

- The following command shows the attacker status in the SI delay calculation report output (when `si_delay_enable_report` attribute is set to `true`):

```
report_delay_calculation -si -from seg3/u5/Y -to seg3/u6/A

From pin : seg3/u5/Y
Cell : INV
```

```
Library : cell_w
To pin : seg3/u6/A
Cell : INV
Library : cell_w
Base or SI: SI delay
Delay type: net

-------------------------------------------------------------
RC Summary for net seg3/n5
-------------------------------------------------------------
Number of capacitance : 17
Net capacitance : 0.293534 pF
Total rise capacitance: 0.320849 pF
Total fall capacitance: 0.320780 pF
Number of resistance : 17
Total resistance : 567.671387 Ohm
-------------------------------------------------------------
SI information for rise
-------------------------------------------------------------------------------------
Victim timing window : 1.0743 ns 2.7024 ns CLK_W_3
-------------------------------------------------------------------------------------
Attacker Status Direction Slew Coupling cap Fanout Voltage Bump Timing window
Clock
-------------------------------------------------------------------------------------
seg2/n5 INF Fall 0.303667 ns 0.138759 pF 1 1.080 v 0.198396 v 1.1917 ns 2.8626 ns
CLK_W_2
seg1/n5 INF Fall 0.314333 ns 0.138787 pF 1 1.080 v 0.182412 v 1.3881 ns 3.0723 ns
CLK_W_1
-------------------------------------------------------------------------------------
SI information for fall
-------------------------------------------------------------------------------------
Victim timing window : 1.2016 ns 5.0700 ns CLK_W_3
-------------------------------------------------------------------------------------
Attacker Status Direction Slew Coupling cap Fanout Voltage Bump Timing window
Clock
-------------------------------------------------------------------------------------
seg2/n5 INF Rise 0.550667 ns 0.138759 pF 1 1.080 v 0.138780 v 1.0887 ns 2.0217 ns
CLK_W_2
seg1/n5 INF Rise 0.800167 ns 0.138787 pF 1 1.080 v 0.107892 v 1.2751 ns 2.3031 ns
CLK_W_1
-------------------------------------------------------------------------------------
Model types
-------------------------------------------------------------------------------------
```

```
Net Term Cell Model
-------------------------------------------------------------------------------
seg3/n5 seg3/u5/Y INV NLDM
seg3/n5 seg3/u6/A INV NLDM
seg2/n5 Y INV NLDM
seg1/n5 Y INV NLDM
-------------------------------------------------------------------------------
Delays Rise Fall
-------------------------------------------------------------
Incr Net Delay : 0.565900 ns 0.093700 ns
Net delay (mean) : 0.696000 ns 0.188900 ns
(3-sigma) : 0.000000 ns 0.000000 ns
Timing Derate (mean) : 1.000000 1.000000
(sigma) : 1.000000 1.000000
Derated Net delay (mean) : 0.696000 ns 0.188900 ns
(3-sigma) : 0.000000 ns 0.000000 ns
-------------------------------------------------------------

-------------------------------------------------------------
Transitions Rise Fall
-------------------------------------------------------------
Input transition (mean) : 0.482600 ns 0.234800 ns
(sigma) : 0.000000 ns 0.000000 ns
Output transition (mean) : 0.622200 ns 0.274100 ns
(sigma) : 0.000000 ns 0.000000 ns
-------------------------------------------------------------
Abbreviative attacker status and filtered reason:
-------------------------------------------------------------
SY : Synchronous
INF: Attacker with infinite timing window
PE : Filtered due to physical exclusion
UF : Filtered by user
CN : Filtered because it is a constant net
SB : Filtered due to small bump
TA : Filtered because of timing window does not overlap
LC : Filtered due to logical correlation
ACT: Attacker is Active.
-------------------------------------------------------------
```

Related Information

- report_timing

reset_decap_cell_candidates

```
reset_decap_cell_candidates
[-help]
```

Clears all available cells for decoupling capacitance insertion. You can use this command to clear previously defined decoupling cell candidates specified using the add_decap_cell_candidates command.

Parameter

`-help`	Outputs a brief description of the `reset_decap_cell_candidates` command. For a detailed description of the command, use the man command: man `reset_decap_cell_candidates`.

Example:

The following command clears all available decoupling cell candidates:

```
reset_decap_cell_candidates
```

Related Information

add_decap_cell_candidates

report_decap_cell_candidates

write_exclude_nets

```
write_exclude_nets
[-help]
out_file
```

Writes out the excluded nets to an output file. Nets that exceed the default value of 1,000 pins are excluded from delay calculation, and default delay values are used. .

You can use `delaycal_excluded_net_file` attribute to use the file containing nets to be excluded during delay calculation. The list of excluded nets can be redirected to an output file.

Parameters

`-help`	Prints out the command usage
`out_file`	Specifies the name of the excluded net file.

Related Information

- `delete_exclude_net_attribute`

write_set_load

```
write_set_load
[-help]
out_file
[-direct]
[-exclude_zero_cap]
[-include_pin_cap]
[-view string]
[-wire_load]
```

Writes the specified capacitance on the specified ports or nets of the top cell to an output file.

Parameters

`-help`	Prints out the command usage.
out_file	Specifies the output file name.
`-direct`	Refers to the ports directly.
`-exclude_zero_cap`	Excludes the net that has zero net capacitance.
`-include_pin_cap`	Includes the pin capacitance.
`-view` *string*	Specifies the view name for the output file.
`-wire_load`	Marks the `set_load -wire_load` option for a port. This interprets the specified capacitance as the external wire capacitance of the port.

10

Flip Chip Commands and Attributes

- assign_bumps
- assign_pg_bumps
- assign_signal_to_bump
- check_bond_pad_spacing
- check_bumps
- check_flip_chip_constraints
- check_io_inst_overlap
- copy_bumps
- create_bump
- create_bump_connect_target_constraint
- create_signal_pin
- delete_area_io_fillers
- delete_bump_connect_target_constraints
- delete_bumps
- flip_chip Category Attributes
- get_bond_pad
- get_physical_info
- get_pin_port_number
- gui_show_bump_connections
- place_bond_pad
- place_bond_pad_by_stagger
- read_bump_color_map_file
- read_flip_chip_property
- read_io_update
- read_package
- report_probe_pins
- report_special_routes
- reset_bumps_fixed_assign
- reset_fixed_bond_pad
- reset_probe_pin
- route_flip_chip
- route_point_to_point
- set_bond_pads_to_fixed
- set_bump_master
- set_bumps_fixed_assign
- set_bumps_place_status
- set_probe_pin
- set_special_route_type
- space_bond_pads

- swap_bump_assignment
- unassign_bumps
- update_bump_connect_target_constraints
- update_bump_name
- write_flip_chip_property

assign_bumps

```
assign_bumps
[-help]
[{-area {x1 y1 x2 y2} | -selected }]
[-constraint_file file_name]
[-exclude_regions {x1 y1 x2 y2} ...]
[-multi_bumps_to_multi_pads]
[{[[-pg_nets net_list] | [-exclude_pg_nets net_list]][-pg_insts instance_list]} [-pg_only]]
```

Assigns the bumps closest to the I/O cells, using euclidean distance, and adds bump connection target property onto bump. These bumps can be signal bumps specified in the Verilog netlist or power/ground (PG) bumps specified in the command line. From the 14.1 release, `assign_bumps` adds bump connect target to bumps by default during bump assignment.

This command is for area I/O flip chip designs. Bumps must be created or loaded from the I/O assignment file.

Use this command after the design is placed.

Note: The `assign_bumps` command does not affect any existing assignment. Hence, it will not reassign any existing power/ground assignments.

Support for CLASS BUMP Property

The `assign_bumps` command supports the `CLASS BUMP` of `PORT` in `LEF`. Using the `LEF` 5.7 syntax, you can add a new property in the `PORT` of the hard macro `LEF` called `CLASS BUMP`. With this property, `route_flip_chip` is able to route from these ports to the assigned bump.

The `assign_bumps` command supports I/O pads or hard macros that have pins with multiple geometries. For the following hard macro definition in LEF, `assign_bumps` assigns three bumps for `block1`, including two bumps for pin `signal1` and one for pin `powerpin`.

:

```
    MACRO block1
        PIN signal1
            PORT
                CLASS BUMP ;
                LAYER … ;
                RECT … ;
            END
            PORT
                CLASS BUMP ;
                LAYER … ;
                RECT … ;
            END
        END signal1
        PIN powerpin
            PORT
                CLASS BUMP ;
                LAYER … ;
                RECT … ;
            END
        END powerpin
```

```
END block1
```

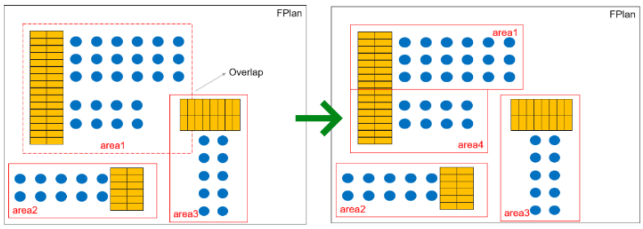
`route_flip_chip` also supports routing for this kind of assignment. In the above example, routing happens from the two signal bump to the two port geometries and one power bump to the powerpin geometry with `CLASS BUMP` property.

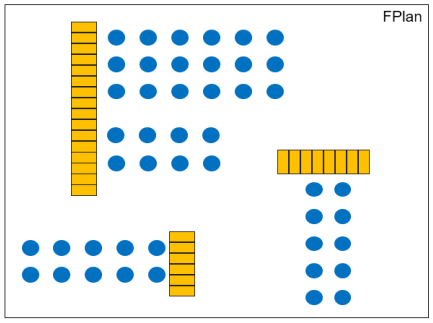
The behavior of `CLASS BUMP` is summarized as below:

- If the `CLASS BUMP` property is not defined in any cell, `assign_bumps` assigns one bump to each IO pin and `route_flip_chip` works normally.
- If any pin port in `CLASS PAD AREAIO` or `CLASS BLOCK` cells contains the `CLASS BUMP` property, `assign_bumps` uses the property for assignment.
- Each pin port with the `CLASS BUMP` property will have a bump assigned. This means a pin can have multiple bumps assigned to it if it has more than one port defined as CLASS `BUMP`. `route_flip_chip` honors this property and routes the port and its assigned bump. A port with `CLASS BUMP` attribute with multiple geometries is considered as one object for assignment and routing.

Parameters

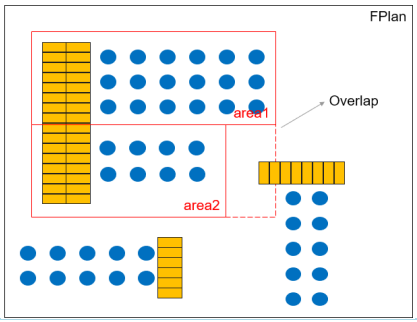
`-help`	Outputs a brief description that includes type and default information for each `assign_bumps` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man assign_bumps`

`-area {x1 y1 x2 y2}`	Assigns objects in the area specified by the *x1* , *y1* , *x2* , and *y2* coordinates.
	• If the specified area includes only bumps, `assign_bumps` assigns the selected bumps to IO cells based on the total shortest distance.
	• If the specified area includes only IO cells, `assign_bumps` assigns the selected IO cells to bumps based on the total shortest distance.
	• If the specified area includes both IO cells and bumps, then `assign_bumps` assigns the selected IO cells to the selected bumps. If the number of bumps are not enough, `assign_bumps` issues a warning.
	Note: The selected objects must be same as the ones selected by the command `gui_window_select`.
	For designs with double pad rows, the following conditions must be kept in mind while defining area with `assign_bumps -area`:
	• All IO pads inside the area should be regular and should be placed in either two rows or two columns. If the pads are irregular, the tool will turn off double pad row assignment automatically and use the traditional optimization algorithm, based on minimal global length.
	• Define areas in such a way that there is only one double pad row in one area.
	• If two areas overlap, it means there are now two double pad rows in one area. The results of bump assignment in such situations might not be the most optimal. To get the most optimal results, split one of the areas into two to avoid overlap as shown below:
	 
	Now, you can assign bumps using the following commands:
	`assign_bumps -area area1`
	`assign_bumps -area area2`
	`assign_bumps -area area3`
	`assign_bumps -area area4`
	Note that for assigning bumps in a design with single pad rows, you can use `assign_bumps` without the `-area` option.



In the following example, the double pad area should be split to avoid overlaps as shown below. The rest of the bumps can be assigned without using the `-area` option:

```
assign_bumps -area area1
assign_bumps -area area2
assign_bumps
```



`-constraint_file` *file_name*	Specifies the path to the file containing bump assignment constraints. At present, the following types of bump assignment constraints are supported: • `SHARE_FIND_PORT` constraint to filter unnecessary ports • `ASSIGN_ANALOG_PG_NETS` constraint to specify which signal nets are analog PG nets • `SHARE_IGNORE_*` and `ASSIGN_IGNORE_*` constraints to exclude instances, macros, pins, or nets for assignment. • `ASSIGN_PAD2BUMP_RATIO` constraint to specify the pad to bump ratio per net, macro, or instance. For more information on specifying bump assignment constraints, refer to the "Using Bump Assignment Constraints" section of the Flip Chip Methodologies chapter of the *User Guide.* *Data_type*: string, optional
`-exclude_pg_nets` *net_list*	Specifies the names of the power/ground nets that must be excluded from bump assignment. Wild cards are supported in net names. The net names should be enclosed in double quotation marks (" ") or braces {}. *Data_type*: (net)*, optional
`-exclude_regions {`*x1 y1 x2 y2 ...*`}`	Excludes bumps in the area specified by the box coordinates. You can specify multiple regions, which may be disjoint. *Data_type*: rect, optional
`-multi_bumps_to_multi_pads`	Specifies that multiple pads need to be assigned to multiple bumps. This parameter is useful when you need to assign multiple pads on one net to multiple bumps based on shortest distance. *Data_type*: bool, optional
`-pg_insts` *instance_list*	Specifies the names of power/ground instances to which the power bumps are assigned. The instance names should be enclosed in double quotation marks (" ") or braces {}. *Data_type*: (inst)+, optional

`-pg_nets net_list`	Specifies the names of the power/ground nets to which the bumps are assigned. The net names should be enclosed in double quotation marks (" ") or braces {}. By default, bumps are assigned to all pads whose ports are connected to power/ground nets. *Data_type*: (net)+, optional
`-pg_only`	Assigns only power/ground nets to bumps. When this option is specified, signals are not assigned to bumps. **Note:** Option `-pg_insts` or `-pg_nets| -exclude_pg_nets` is required for option `-pg_only`. *Data_type*: bool, optional
`-selected`	Assigns selected bumps to the closest I/O instances or selected I/O instances to the closest bumps. • If only bumps are selected, `assign_bumps` assigns the selected bumps to IO cells based on the total shortest distance. • If only IO cells are selected, `assign_bumps` assigns the selected IO cells to bumps based on the total shortest distance. • If both IO cells and bumps are selected, `assign_bumps` assigns the selected IO cells to the selected bumps. If there are not enough bumps, `assign_bumps` issues a warning. **Note:** The `-selected` parameter works with other options of `assign_bumps`, except `-area` and `-exclude_regions`. *Data_type*: bool, optional

Examples

- The following command assigns bumps to the power and ground nets, `VDD` and `VSS`, within the area specified by the coordinates `600.0`, `960.0`, `2600.0`, and `4100.0`.
  ```
  assign_bumps -pg_nets {VDD VSS} -area {600.0 960.0 2600.0 4100.0}
  ```
- The following command assigns multi-PG pads to bumps:
  ```
  assign_bumps -pg_only -pg_nets {VDD VSS}
  ```

Related Information

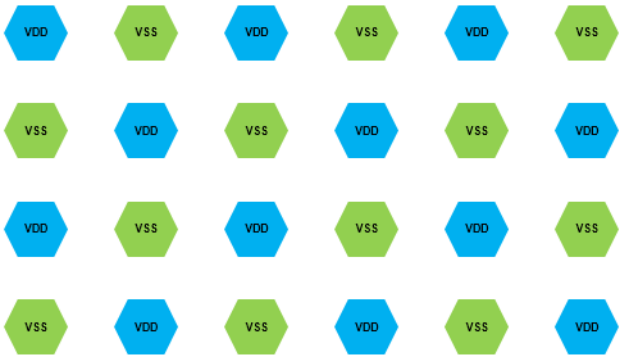
- `assign_pg_bumps`
- `assign_signal_to_bump`
- `unassign_bumps`
- Flip Chip Methodologies chapter of the *User Guide.*
- Tools Menu chapter of the *Innovus Menu Reference*
  - *Flip Chip*

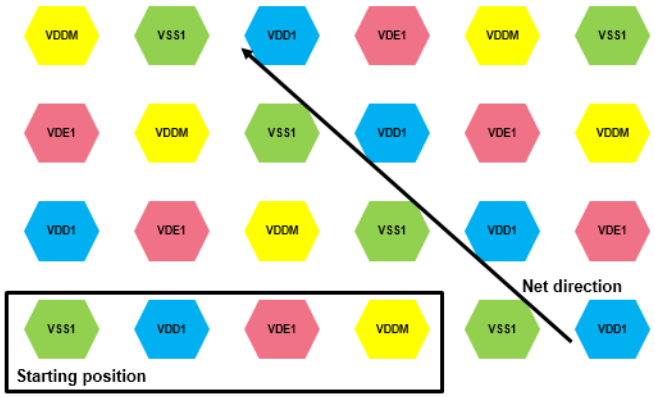
assign_pg_bumps

```
assign_pg_bumps
[-help]
[-connect_type {ioring | corering | stripe | iopin}]
-nets {nameList}
[-selected | -floating | -bumps {bump_name_list}]
[-vertical | -horizontal | -checker_board [-square {width height}]]
```

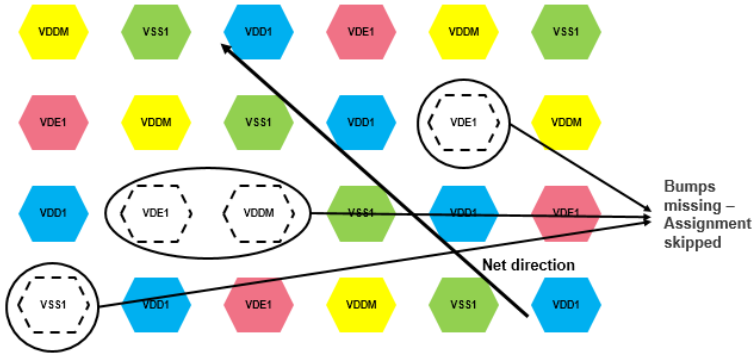
Assigns power and ground bumps to connect flip chip I/O pins and adds property value onto bumps. Use this command after bumps are created. It overwrites existing assignments for selected bumps.

Parameters

`-help`	Prints a brief description that includes type and default information for each `assign_pg_bumps` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man assign_pg_bumps`
`-bumps {bump_name_list}`	Specifies the names of the bumps to be assigned. The names must be enclosed in braces. Wildcards are supported in bump names. **Note**: This parameter cannot be used with `-selected` and `-floating`. *Data_type*: (bump)+, required
`-checker_board`	Specifies that the checkerboard style of bump assignment will be used if multiple nets are defined using the `-nets` parameter:  The `-checker_board` parameter is typically used for a regular (rectangular) bump array. To apply this pattern for an irregular (rectilinear) bump array, first form a regular array by creating virtual bumps in the areas where there are no bumps. Next, assign the bumps using the checkerboard pattern and then remove the virtual bumps. **Notes** • The checkerboard style can be used with more than two nets, as shown below: `assign_pg_bumps -nets {VSS1 VDD1 VDE1 VDDM} -checker_board -selected`

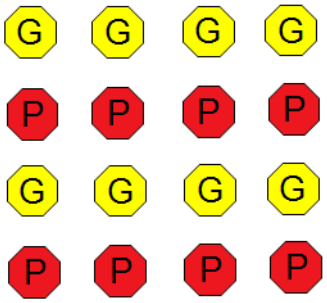
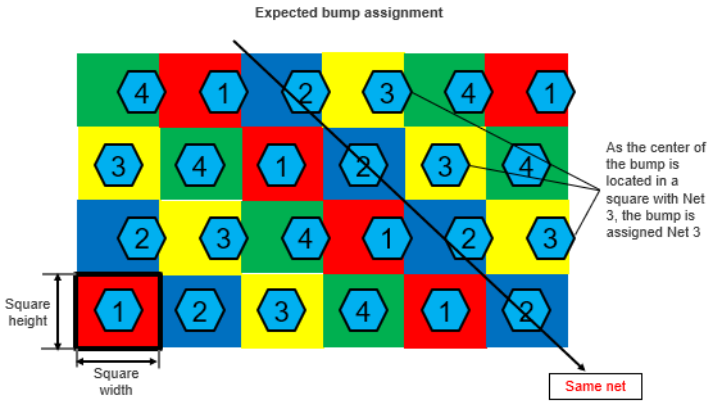


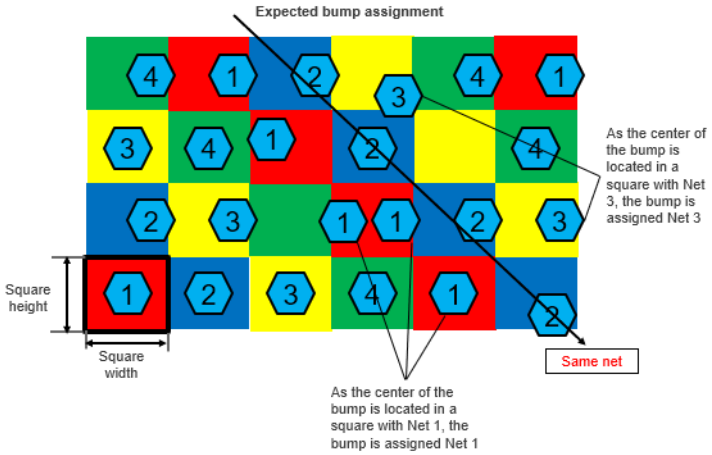
The `-checker_board` option for multiple nets can be used in a regular bump matrix with the same bump cell for all bumps and the same X/Y pitch between bumps. If bumps are missing in the pattern, the tool skips assignment to maintain the diagonal, staggered net direction.



- The checkerboard style can support irregular bump assignment when used with the `-square` parameter. For details, see the description of the `-square` parameter.

`-connect_type {ioring | corering | stripe | iopin}`	
	Specifies the bump connect type when assigning power/ground bumps. Connection type can be I/O Ring, Core Ring, Stripe, or I/O Pin. *Data_type*: enum, optional
`-floating`	Assigns to all floating bumps. *Data_type*: bool, required

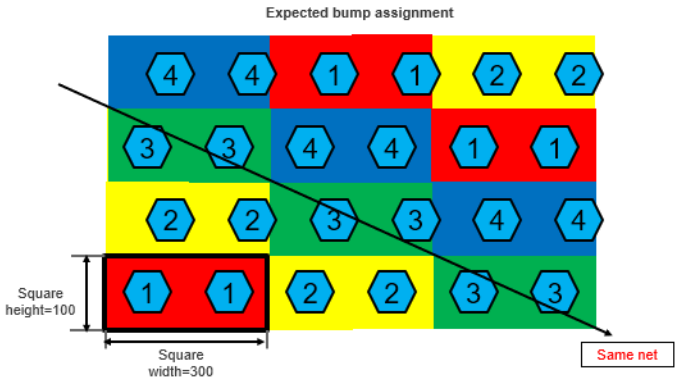
`-horizontal`	Specifies that the horizontal style of bump assignment will be used if multiple nets are defined using the `-nets` parameter:  *Data_type*: bool, required
`-nets {nameList}`	Specifies the net names. Net names must be enclosed in braces. *Data_type*: (net)+, required
`-selected`	Assigns to selected bumps. *Data_type*: bool, required
`-square {width height}`	Specifies the width and height of a square in the checkerboard style of bump assignment. Use this parameter if you want to apply the checkerboard style of bump assignment for an irregular bump matrix. When you specify the `-square` parameter, the tool creates a virtual checkerboard made of squares of the specified dimensions and then assigns the squares with the specified nets. The square to which a bump belongs is determined based on the location of its center. The tool then assigns the corresponding net to the bump. For example, the following command creates a virtual checkerboard with squares of width 150 and height 100, assigns the squares with the Nets 1, 2, 3, and 4, and assigns the nets to bumps based on the square on which their center is located: `assign_pg_bumps -nets {1 2 3 4} -checker_board -selected -square {150 100}` **Case A**  As shown in Case B below, the assignment is dependent on the location of the center of the bumps. **Case B**



The assignment also changes if you change the dimensions specified for the checkerboard squares. For example, the following command creates a virtual checkerboard with squares of width 300 and height 100.

```
assign_pg_bumps -nets {1 2 3 4} -checker_board -selected -square {300 100}
```

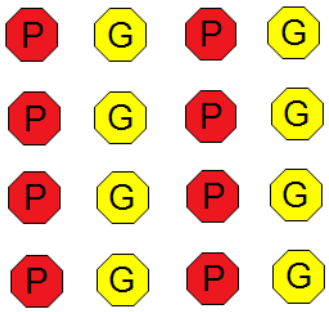
Case C



Note that although the bump locations are same in both Cases A and C, the assignments in Case C are different from Case A

`-vertical`

Specifies that the vertical style of bump assignment will be used if multiple nets are defined using the `-nets` parameter:



Data_type: bool, required

Example

- The following command assigns selected bump to power net `VDD`:

  ```
  assign_pg_bumps -nets VDD -selected
  ```

- The following command assigns only the specified bumps to the power net `VDD`:

  ```
  assign_pg_bumps -nets VDD -bumps {Bump_1 Bump_2 Bump_3}
  ```

- The following command assigns floating bumps to the specified nets in the horizontal style:
  ```
  assign_pg_bumps -horizontal -floating -nets {VSS1 VDD1 VDE1}
  ```
- The following command assigns selected power/ground bumps in the checkerboard style to the VDD and VSS nets:
  ```
  assign_pg_bumps -selected -nets {VDD VSS} -checker_board
  ```
- The following command assigns all the floating bumps in checkerboard pattern to the VDD and VSS nets:
  ```
  assign_pg_bumps -floating -nets {VDD VSS} -checker_board
  ```

Related Information

- `assign_bumps`
- `assign_signal_to_bump`
- `unassign_bumps`
- Flip Chip Methodologies in the *User Guide*
- Tools Menu chapter in the *Menu Reference*
  - *Flip Chip*

assign_signal_to_bump

```
assign_signal_to_bump
[-help]
{ {{-net net_name | {-top_pin port [-bump_pin bump_pin_name]}} {-bumps bump_name_list | -selected}} }
```

Assigns selected or specified bumps to the specified net or pin. The command supports signal, power, as well as ground nets. Feedthrough bumps with multiple pins are also supported.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `assign_signal_to_bump` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man assign_signal_to_bump`
`-bump_pin bump_pin_name`	Specifies the name of the bump pin that has to be assigned to the specified top pin. Use this option for feedthrough bumps, which have multiple pins. This parameter can be used only with the `-top_pin` parameter. *Data_type*: string, optional
`-bumps bump_name_list`	Specifies the bumps to be assigned to the net specified with `-net`. Use braces to enclose the bump names. *Data_type*: string, optional
`-net net_name`	Specifies the signal or PG net to which bumps are to be assigned. *Data_type*: (net), optional
`-selected`	Assigns the selected bump to the specified nets. *Data_type*: bool, optional
`-top_pin port`	Specifies the name of the top-level pin to which bumps are to be assigned. After assignment, the pin name of the bump is the top-level pin name and the net name of the bump is the net name of the specified pin. This parameter cannot be used with the `-net` parameter. *Data_type*: string, optional

Examples

- The following command assigns `bump_1` and `bump_2` to the signal net `sig[1]`:
  ```
  assign_signal_to_bump -net sig[1] -bumps {bump_1 bump_2}
  ```
- The following command assigns `bump_1` to the power net `VDD`:
  ```
  assign_signal_to_bump -net VDD -bumps bump_1
  ```
- The following command assigns the selected bumps to the ground net `VSS`:
  ```
  assign_signal_to_bump -net VSS -selected
  ```
- The following command assigns the `PIN_1` pin of the `bump1` feedthrough bump to the top-level pin, `CLK_IN`:
  ```
  assign_signal_to_bump -bumps bump1 -top_pin CLK_IN [-bump_pin PIN_1]
  ```

Related Information

- `assign_bumps`
- `assign_pg_bumps`
- `unassign_bumps`
- Flip Chip Methodologies in the *User Guide*
- Tools Menu chapter in the *Menu Reference*
  - *Flip Chip*

check_bond_pad_spacing

```
check_bond_pad_spacing [-help]
```

Checks for bond pad spacing violations after using the `space_bond_pads` command. You can use this command at any time after using the `space_bond_pads` command.

Parameters

`-help`	Prints a brief description of the `check_bond_pad_spacing` command. For a detailed description of the command, use the `man` command: `man check_bond_pad_spacing`

Related Commands

- `get_bond_pad`
- `place_bond_pad`
- `place_bond_pad_by_stagger`
- `reset_fixed_bond_pad`
- `set_bond_pads_to_fixed`
- `space_bond_pads`

check_bumps

```
check_bumps
[-help]
[-bump_to_pin_overlap inst+]
[-delete_markers]
[-out_file fileName]
[-select_violation_bumps]
[-selected | -region {x1 y1 x2 y2}]
[-relative_type {embedded_bump inst_pin_port} [[-relative_obj {inst_pin_port_list}] [-relative_assignment ] [-relative_offset x y]]]
[ [[-relative_type {embedded_bump inst_pin_port}] [-bump_to_pin_pitch {pitch_value {[bump_name] [inst_pin_name [pin_x pin_y]]}}]]
  | -nets net+ ]
[-bump_to_edge_pitch pitch_value [-edge_offset {offset [corner_offset]}]]
[ [[-bump_pitch {pitch_value [bump_name]}]
   [-bump_to_pin_pitch {pitch_value {[bump_name] [inst_pin_name [pin_x pin_y]]}}]]
  | -bump_max_pitch max_pitch_value | -bump_to_edge_pitch pitch_value]
[-on_bump_grid [-on_bump_grid_tolerance bump_grid_name value]]
```

Checks the legality of the bump assignment and creates a report, which is either displayed in the console window or written to a file. Use this command for bump overlap and pitch checking. The `check_bumps` command can be used after bumps are assigned with the `assign_bumps` command or defined in the I/O file with the signal name but without the `-fixed` constraint.

Bump-to-Bump Overlap Checking

`check_bumps` checks bump overlaps, including overlap of dummy bumps, by default. If a bump overlaps with others, `check_bumps` places violation markers on the center of the overlapped bumps and reports the total number of overlapped bumps.

The violation markers for bump overlap can be browsed in Violation Browser.

⚠ `check_bumps`:

- Checks overlaps between regular bumps by checking for pin overlaps on bumps.
- Checks overlaps between dummy bumps by checking for OBS overlap.
- Checks overlaps between dummy bumps and regular bumps by checking for OBS and bump pin overlap.

Bump-to-Pin Overlap Checking

If `-bump_to_pin_overlap` is used, `check_bumps` checks for overlaps between bump shapes, including dummy bumps, and pins of the specified block. If the bump shape and the block pins are at the same location but on different layers, the tool does not consider them to be overlapping. However, if the bump shapes and the block pins are at the same location and on the same layer, they are reported as overlapping, as shown below:

Bump Pitch Checking

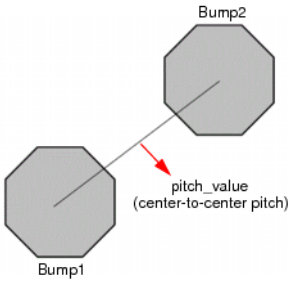
`check_bumps` provides a bump pitch checking feature to record bumps that violate the bump pitch. The bump pitch checking feature has multiple options:

- `-bump_pitch` for checking the center-to-center pitch between bumps. The bumps that have a smaller center-to-center distance than the specified *`pitch_value`* are marked as spacing violations.
- `-bump_max_pitch` for checking whether the center-to-center pitch between neighboring bumps violates the maximum spacing constraint.
- `-bump_to_edge_pitch` for checking the pitch from the bump center to the guard or die edge.
- `-bump_to_pin_pitch` for checking the pitch between bump and pins of hard macros. The bumps and instance pin pairs that have a smaller bump_center-to-pin_center or bump_center-to-specified_pin_geometry_location distance than the *`pitch_value`* are marked as spacing violations.

In all these cases, bumps that violate the pitch request are marked as spacing violations and can be checked using the Violation Browser.

Bump-to-Bump Pitch Checking

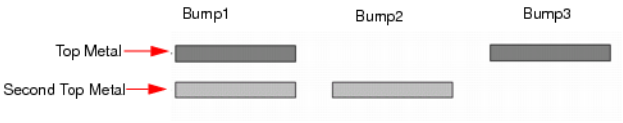
For bump to bump pitch checking, the center-to-center bump pitch is needed. The center of a bump is calculated using the bump cell geometry at the topmost RDL layer.



Note: Only bump cell with single geometry at one RDL layer is supported. For multiple-layer bump cell, the topmost layer in the bump cell is the topmost RDL layer. If no geometry is defined at that layer, the geometry on the second topmost layer is used and a warning is issued:

No geometry at XX layer found; the geometry at XXX layer is used for bump center calculation.

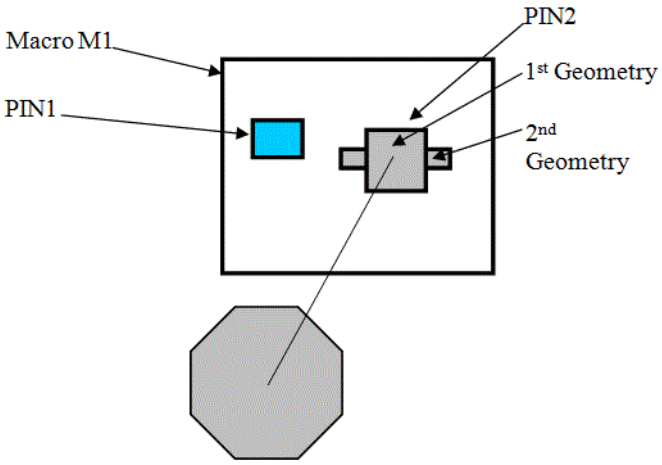
If the check is between a multiple-layer bump and a single-layer bump, the center is calculated using the geometries on the topmost layer in each cell. For example, in the following diagram, center-to-center pitch between `Bump1` and `Bump2` is calculated using the geometries on the top metal in `Bump1` and the second top metal layer in `Bump2`.



If there is no common layer in two bumps, there is no center-to-center pitch to be checked. For example, there is no violation to be reported for `Bump2` and `Bump3` in the above diagram.

Bump-to-Pin Pitch Checking

In some designs with IP macros, bumps have already been embedded inside the macro and are considered as macro pins. In this case, `check_bumps` checks pitch between the bump center and the pin center. Bump center is defined in the same way as in the bump-to-bump case. Pin center calculation for pitch checking can vary as follows:

- Pin has multiple geometries on common layer
  `check_bumps` selects all geometries of the pin if its layer is equal to bump's layer. For example, if bump only has topmost geometry, `check_bumps` selects all the geometries on the topmost layer for pin. If the bump has both topmost and second layer geometry, `check_bumps` selects the geometries for both layers. If no valid geometry of pin is found, `check_bumps` skips the pin pitch checking for that pin.
  In the example below, the tool checks one by one the pitch for each geometry of `PIN2` on `Bump1`'s layer to see if there is a violation. That is, `check_bumps` calculates the distance from the center of 1st geometry of `PIN2` to the center of `Bump1`. Then, it calculates distance from the center of 2nd geometry of `PIN2` to center of `Bump1`. For each check, `check_bumps` reports a violation if the distance is less than `100`.
  ```
  check_bumps -bump_to_pin_pitch {100 {Bump1 M1/PIN2}}
  ```
  

- Pin has multiple geometries and a particular geometry location is specified
  In case of multiple geometries, `check_bumps` allows you to specify a location `[pin_xpin_y]`. If this location overlaps with a geometry, the pitch to this geometry is checked. `check_bumps` calculates the distance from the center of this geometry to the bump center.

Note: `check_bumps` does not support pin-to-pin pitch checking because such violations can only be fixed at LEF level which is beyond the flipchip tool capability. So in the following command:

```
check_bumps -bump_to_pin_pitch {100 {Bump1 M1/PIN1 M1/PIN2}}
```

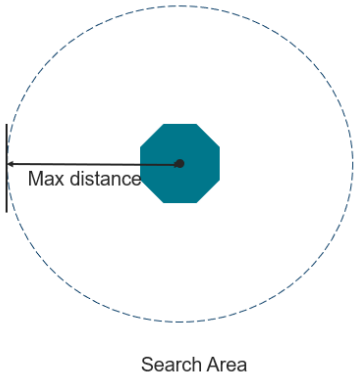
`check_bumps` checks `Bump1` and `M1/PIN1`; and `Bump1` and `M1/PIN2`; but not `M1/PIN1` and `M1/PIN2`.

Parameters

`-help`	Prints a brief description that includes type and default information for each `check_bumps` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man check_bumps`

`-bump_max_pitch max_pitch_value`

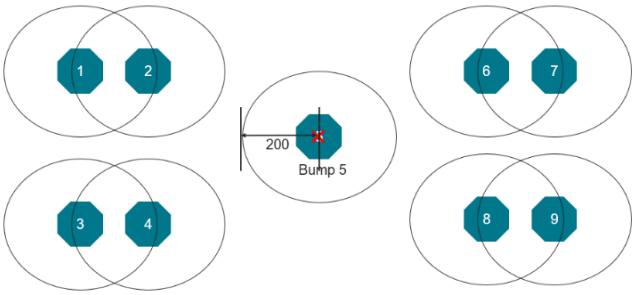
Checks the center-to-center pitch between neighboring bumps in microns. For any bump, the search area is defined by a circle wi of `max_pitch_value` and the bump center as its center.



If another bump is detected within a bump's search area, the bump is DRC free. If no other bump is detected in a bump's search a bump is marked with a maximum spacing violation and a warning message is displayed.

In the example below, only Bump 5 violates the maximum spacing constraint and is flagged with a maximum spacing violation:

```
check_bumps -bump_max_pitch 200
```



The `-bump_max_pitch` parameter works on:

- all bumps in the design by default.
- selected bumps if `-selected` is specified.
- bumps in a specific area if `-region` is specified.
- same-net bumps if `-nets` is specified.

Note: The `-bump_max_pitch` parameter cannot be specified along with the `-bump_pitch` or the `-bump_to_edge_pitch` parameter.

Data_type: float, optional

`-bump_pitch {pitch_value [bump_name]}`

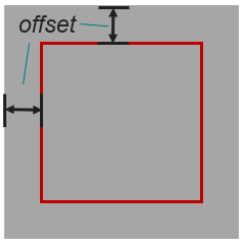
	Checks the center-to-center pitch between bumps. Here: • *pitch_value* represents the center-to-center pitch value in microns. • *bump_name* is the name of the bump. You can check pitch between two or more bumps. Wildcards can be used to specify bump If more than two bumps are specified, bump pitch is checked for all possible pairings. In the following example, check_bumps ch center-to-center pitch between B1 and B2; B1 and B3; B2 and B3: check_bumps -bump_pitch {100 {B1 B2 B3}} Any illegal pitch is logged as "Bump pitch between B1 and B2 XX < 100." Specifying bump names is optional. If bump names are specified, -bump_pitch works on the specified bumps. If bump names a specified, -bump_pitch works on: ◦ selected bumps if -selected is specified. ◦ bumps in a specific area if -region is specified. ◦ all bumps in the design, if neither -selected nor -region is specified. **Note:** You can specify pitch checking multiple times in this option. *Data_type*: string, optional
-bump_to_edge_pitch *pitch_value*	
	Checks whether the pitch from the bump center to the guard or die edge is within the specified pitch value. If the detected pitch is l the specified value, a warning is displayed and the bump is flagged with a spacing violation. *pitch_value* is specified in microns. By default, if -bump_to_edge_pitch is specified, the tool checks the pitch from the bump center to the die edge. You can additional edge_offset with this parameter to define a guard shape. If -edge_offset is specified, the tool checks the distance from the bump the guard edge defined by -edge_offset. If the bump center is outside the guard edge, the center-to-guard pitch will be a negative number, which is not allowed. The tool is warning in such cases. Check the **Examples** section to see how the bump_to_edge_pitch parameter can be used. The -bump_to_edge_pitch parameter works on: • all bumps in the design by default. • selected bumps if -selected is specified. • bumps in a specific area if -area is specified. • bumps with specific nets if -nets is specified. **Note**: The -bump_to_edge_pitch parameter cannot be specified along with the -bumpPitch or the -bumpMaxPitch parameter. *Data_type*: float, optional
-bump_to_pin_overlap *inst+*	
	Checks if the bump overlaps with the pins of the specified block. If the bump shape and the block pins are at the same location bu different layers, the tool does not consider them to be overlapping. However, if the bump shapes and the block pins are at the sam and on the same layer, they are reported as overlapping. The -bump_to_pin_overlap parameter works on: • all bumps, by default. • selected bumps, if -selected is specified. • bumps in the specified area, if -region is specified. • bumps with the specified nets, if -net is specified. *inst+* should be of type block. Overwise, the tool reports an error. *Data_type*: (inst)+, optional
-bump_to_pin_pitch {*pitch_value* {[*bump_name*][*inst_pin_name* [*pin_x pin_y*]]}}	

	Checks the center-to-center pitch between bumps and pins of hard macros. Here: • *pitch_value* represents the center-to-center pitch value in microns. • *bump_name* is the name of the bump. • *inst_pin_name* is the name of the instance pin. • *pin_x* and *pin_y* specify the pin geometry. **Note:** You can specify pitch checking multiple times in this option. Wildcards can be used in bump and pin names. *Data_type*: string, optional
-delete_markers	Resets violation markers. *Data_type*: bool, optional
-edge_offset {*offset* [*corner_offset*]}	

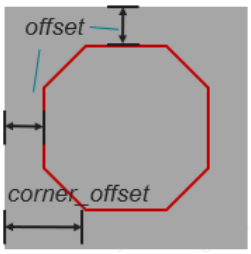
Defines the guard shape if it is different from the die boundary. The `-edge_offset` parameter can be used only if the `-bump_to_edg` parameter is specified.

You can specify either one or two values with the `-edge_offset` parameter:

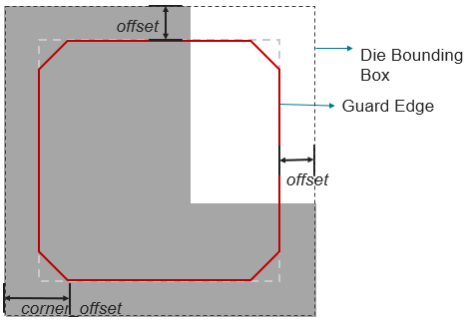
- If only one value (*`offset`*) is specified, the guard edge shape is a rectangle with the die boundary to guard edge distance being the specified `offset` value.



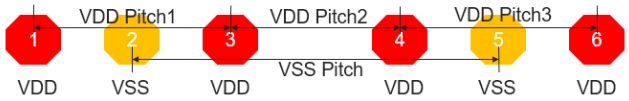
- If two values (*`offset`* and *`corner_offset`*) are specified, the guard edge shape is an octagon with 45-degree exterior angles. H guard edge to the die boundary distance is the specified *`offset`* value and the distance from the die boundary to the lower-left guard edge is the specified *`corner_offset`* value.



In a rectilinear design, you can use the bounding box of the die as a reference for specifying the offset for the guard edge.



Data_type: string, optional

`-nets net+`	Checks constraints for the bumps with the specified nets. If `-bump_max_pitch` is specified along with `-nets`, the tool checks for the maximum pitch constraint only between bumps belonging same net. For instance, in the example given below, if you specify the command `check_bumps -bump_max_pitch 300 -nets {VDD` tool checks: • Checks the `VDD Pitch1` between bump1 and bump3, `VDD Pitch2` between bump3 and bump4, and `VDD pitch3` between bump bump6 • Checks the `VSS Pitch` between bump2 and bump5.  If `-bump_max_pitch` is not specified, the tool performs default checks for the bumps with the specified nets. **Note**: The `-nets` parameter cannot be used along with the `-bump_max_pitch` or `-relative_type` parameter. *Data_type*: (net)+, optional
`-on_bump_grid`	Checks if the bumps are on bump grids. This option cannot be used with any of the other parameters, except `-on_bump_grid_tole` and `-out_file`. The command errors out in the following cases: • If there is no bump grid in the DB - In this case, use the `create_bump_grid` command to create a bump grid. • If there are multiple bump grids on a single layer The command reports an error but keeps checking in the following case: • If there are multiple bump grids in different layers for one bump Warnings are reported in the following cases: • If there is no bump grid on the bump's layer • If a bump is off bump grid; the bump is also marked with a DRC marker. *Data_type*: bool, optional
`-on_bump_grid_tolerance bump_grid_name value`	
	Specifies the tolerance value to be used when checking if bumps are on the specified bump grid. If this parameter is specified, the checks the Euclidean distance between the bump center and the specified grid and reports a violation only if the distance is more specified tolerance value. ⚠ • The `-on_bump_grid_tolerance` option can be specified only when `-on_bump_grid` is specified. • The value specified with `-on_bump_grid_tolerance` should be smaller than half of the bump grid. If not, the tool returns an *Data_type*: string, optional
`-out_file fileName`	Specifies the report filename. If no filename is specified, the report is printed to the console window. *Data_type*: string, optional
`-region {x1 y1 x2 y2}`	Checks constraints for the bumps in the specific area. The unit is microns. The bumps selected by `-region` should be the same a selected by the command `gui_select`. If `-region` is used with `-bump_pitch`, `check_bumps` checks the bump pitch only for the bumps in the specific area. If `-region` is spec without `-bump_pitch`, `check_bumps` performs the default checks for the bumps in the specific area. **Note:** This parameter cannot be used with with `-bump_to_pin_pitch` or `-selected`. *Data_type*: rect, optional

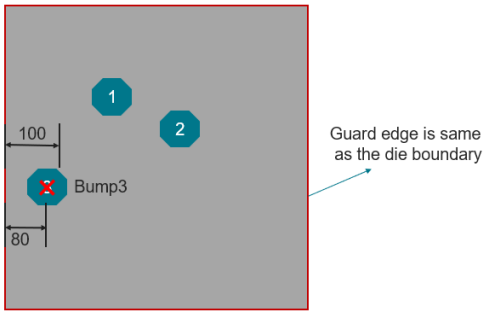
`-relative_assignment`	Checks whether the net assignment of the target bump is equivalent to corresponding relative object. If not, it issues an error mess places a violation marker at the geometry center of the target bumps:
	• For `-relative_type embedded bump`, the following error message is displayed:
	`The assignment of $bump $net is not connected to the pin of the relative object $blockName/$pinName $expectedNe`
	• For `-relative_type inst_pin_port`, the following error message is displayed:
	If `-relative_type` is specified as `embedded_bump` and there is no bump in the region being checked, `check_bumps` does not check fi relative assignment and issues the following error message:
	`There is no bump overlapping with the relative object $blockName/$pinName located at {x2 y2} with relative offset stop checking the relative assignment for this relative object.`
	Note: This option can be used only if `-relative_type` is specified.
	Data_type: bool, optional

`-relative_obj {inst_pin_port_list}`	
	Specifies the inst_pin_port list and supports wild cards. The format is `{inst:pin:port ...}` and `port` is of type integer. If `-relative_object` is specified, `check_bumps` checks whether the geometry center of the target bumps is at the same location as t geometry center of the specified port. If not, it issues an error message and places a violation marker on the geometry center of the bumps. In addition: • If no bump overlaps with the specified port, `check_bumps` displays the message that there are no bumps in the design overlappi specified port. • If the specified ports do not exist, an error message is displayed. **Note**: When used with `-selected` or `-region`, the `relative_object` option works only on the specified bumps by selection or area. *Data_type*: string, optional
`-relative_offset x y`	Specifies the offset of bump location compared with the corresponding relative object. This option can be used only if `-relative_` specified. *Default*: `{0 0}` *Data_type*: point, optional
`-relative_type {embedded_bump inst_pin_port}`	
	Specifies the type of relative object: • `embedded_bump`: If this option is specified, `check_bumps` checks whether the geometry center of the target bumps is at the same l the geometry center of the overlapping embedded bumps. If not, it issues an error message and places a violation marker on th center of the target bumps. In addition: ◦ If no bump overlaps with the embedded bumps, `check_bumps` displays an error message as follows: `There is no bump overlapping with the relative object $blockName/$pinName located at {x2 y2} with relative o y}.` ◦ If more than one bump is located at the region being checked, all bumps violating the embedded bump are reported. A sum number of the bumps violating the embedded bump is displayed. Violation markers are placed on the embedded bump as v bumps violating the embedded bump. The violations can be viewed in Violation Browser. ◦ If there are no embedded bumps, an error message is displayed. **Note**: As this check happens on the entire design, you cannot use the `-selected` or `-region` options if `-relative_typeembedde` specified. • `inst_pin_port`: If the type is `inst_pin_port`, you must specify the `-relative_object {inst_pin_port_list}` option. *Data_type*: enum, optional
`-select_violation_bumps`	
	Selects automatically all the bumps that are violating any check rules.

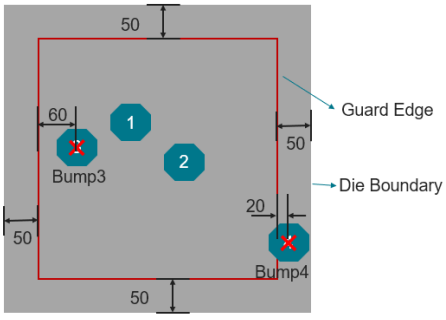
`-selected`	Checks constraints for selected bumps. If `-selected` is used with `-bump_pitch`, `check_bumps` checks the bump pitch only for the selected bumps. If `-selected` is specified w `bump_pitch`, `check_bumps` performs the default checks for the selected bumps. **Note**: This parameter cannot be used with with `-bump_to_pin_pitch` or `-region`. *Data_type*: bool, optional

Examples

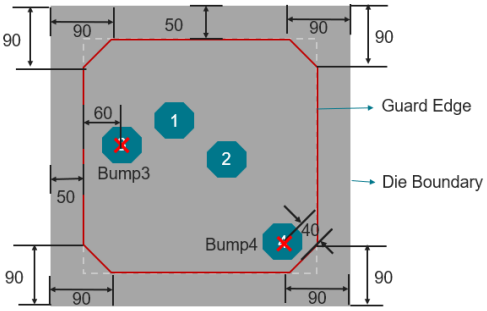
- The following command checks the center-to-center pitch for all bumps. If the center-to-center distance between two bumps is smaller than 100, it reports a violation:
  ```
  check_bumps -bump_pitch 100
  ```
- The following command checks the center-to-center pitch between bump_1 and bump_2:
  ```
  check_bumps -bump_pitch {100 {bump_1 bump_2}}
  ```
- The following command checks the center-to-center pitch for the selected bumps:
  ```
  check_bumps -bump_pitch 100 -selected
  ```
- The following command checks the center-to-center pitch for bumps in the specified area:
  ```
  check_bumps -bump_pitch 100 -region {100 400 600 900}
  ```
- The following command checks whether the bump center to die edge pitch is less than 100 microns:
  ```
  check_bumps -bump_to_edge_pitch 100
  ```
  In the design below, Bump3 is flagged with a spacing violation and a warning message is displayed:



- The following command checks whether the bump center to guard edge pitch is less than 100 microns. The guard edge is offset from the die boundary by 50 microns:
  ```
  check_bumps -bump_to_edge_pitch 100 -edge_offset 50
  ```
  In the design below, Bump3 and Bump4 are flagged with spacing violations and warnings are displayed:



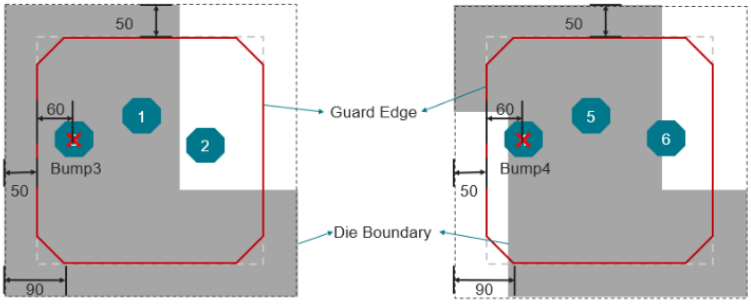
- The following command checks whether the bump center to guard edge pitch is less than 100 microns. The guard edge is offset from the die boundary by 50 microns and the guard corners are offset by 90 microns:
  ```
  check_bumps -bump_to_edge_pitch 100 -edge_offset {50 90}
  ```
  In the design below, Bump3 and Bump4 are flagged with spacing violations and warnings are displayed:



- The following command checks whether the bump center to guard edge pitch is less than 100 microns. The guard edge is offset from the die boundary by 50 microns and the guard corners are offset by 90 microns:

  ```
  check_bumps -bump_to_edge_pitch 100 -edge_offset {50 90}
  ```

  In the rectilinear designs below, Bump3 and Bump4 are flagged with spacing violations and warnings are displayed. However, Bumps2 and Bump6 are DRC free even though they are outside the die boundary because they are still within the guard shape:



- The following command checks the bump assignment and sends the report to `test.rpt`:

  ```
  check_bumps -out_file test.rpt
  ```

  A sample report file is as follows:

```
################################################################
# Generated by: Cadence Innovus 21.10-d227_1
# OS: Linux x86_64(Host ID noivl-leenap)
# Generated on: Wed Dec 2 12:19:39 2020
# Design: DTMF_CHIP
# Command: check_bumps -out_file test.rpt
################################################################

*** The statistics information for bumps ***
Bump_23_2_2 Assigned port_pad_data_out[9] IOPADS_INST/Ptdspop09
Bump_22_1_2 Assigned port_pad_data_out[10] IOPADS_INST/Ptdspop10
Bump_61_0_6 Assigned port_pad_data_out[11] IOPADS_INST/Ptdspop11
Bump_14_3_1 Assigned port_pad_data_out[14] IOPADS_INST/Ptdspop14
Bump_37_6_3 Assigned port_pad_data_in[4] IOPADS_INST/Ptdspip04
Bump_74_3_7 Assigned port_pad_data_in[9] IOPADS_INST/Ptdspip09
Bump_32_1_3 Assigned port_pad_data_in[13] IOPADS_INST/Ptdspip13
Bump_62_1_6 Assigned vcom IOPADS_INST/Pvcomop
Bump_83_2_8 Assigned scan_out_2 IOPADS_INST/Pscanout2op
Bump_12_1_1 Assigned port_pad_data_out[5] IOPADS_INST/Ptdspop05
Bump_66_5_6 Assigned port_pad_data_out[6] IOPADS_INST/Ptdspop06
Bump_4_3_0 Assigned port_pad_data_out[12] IOPADS_INST/Ptdspop12
Bump_51_0_5 Assigned port_pad_data_in[7] IOPADS_INST/Ptdspip07
Bump_13_2_1 Assigned port_pad_data_in[3] IOPADS_INST/Ptdspip03
Bump_71_0_7 Assigned port_pad_data_in[6] IOPADS_INST/Ptdspip06
Bump_44_3_4 Assigned port_pad_data_in[12] IOPADS_INST/Ptdspip12
Bump_89_8_8 Assigned port_pad_data_in[14] IOPADS_INST/Ptdspip14
Bump_52_1_5 Assigned port_pad_data_in[15] IOPADS_INST/Ptdspip15
Bump_26_5_2 Unassigned
Bump_35_4_3 Assigned port_pad_data_out[4] IOPADS_INST/Ptdspop04
Bump_41_0_4 Assigned port_pad_data_out[7] IOPADS_INST/Ptdspop07
Bump_1_0_0 Assigned port_pad_data_out[13] IOPADS_INST/Ptdspop13
```

```
Bump_73_2_7 Assigned port_pad_data_in[5] IOPADS_INST/Ptdspip05
Bump_63_2_6 Assigned port_pad_data_in[8] IOPADS_INST/Ptdspip08
Bump_55_4_5 Assigned port_pad_data_in[11] IOPADS_INST/Ptdspip11
Bump_88_7_8 Assigned pllrst IOPADS_INST/Ppllrstip
Bump_3_2_0 Assigned ibias IOPADS_INST/Pibiasip
Bump_34_3_3 Unassigned
Bump_42_1_4 Unassigned
Bump_72_1_7 Unassigned
Bump_5_4_0 Assigned port_pad_data_out[8] IOPADS_INST/Ptdspop08
Bump_19_8_1 Assigned port_pad_data_out[15] IOPADS_INST/Ptdspop15
Bump_85_4_8 Assigned port_pad_data_in[10] IOPADS_INST/Ptdspip10
Bump_46_5_4 Unassigned
Bump_47_6_4 Assigned refclk IOPADS_INST/Prefclkip
Bump_84_3_8 Assigned reset IOPADS_INST/Presetip
Bump_25_4_2 Unassigned
Bump_48_7_4 Unassigned
Bump_24_3_2 Unassigned
Bump_33_2_3 Unassigned
Bump_11_0_1 Unassigned
Bump_75_4_7 Unassigned
Bump_43_2_4 Assigned spi_data IOPADS_INST/Pspidip
Bump_82_1_8 Assigned spi_fs IOPADS_INST/Pspifsip
Bump_76_5_7 Assigned scan_in_2 IOPADS_INST/Pscanin2ip
Bump_27_6_2 Assigned vcop IOPADS_INST/Pvcopop
Bump_7_6_0 Assigned scan_out_1 IOPADS_INST/Pscanout1op
Bump_6_5_0 Assigned tdigit[0] IOPADS_INST/Ptdigop0
Bump_17_6_1 Unassigned
Bump_79_8_7 Unassigned
Bump_2_1_0 Assigned int IOPADS_INST/Pintip
Bump_64_3_6 Assigned test_mode IOPADS_INST/Ptestmdip
Bump_67_6_6 Assigned scan_clk IOPADS_INST/Pscanckip
Bump_31_0_3 Assigned scan_en IOPADS_INST/Pscanenip
Bump_78_7_7 Assigned tdigit[1] IOPADS_INST/Ptdigop1
Bump_16_5_1 Assigned tdigit[2] IOPADS_INST/Ptdigop2
Bump_10_9_0 Assigned tdigit[3] IOPADS_INST/Ptdigop3
Bump_30_9_2 Assigned tdigit[6] IOPADS_INST/Ptdigop6
Bump_29_8_2 Unassigned
Bump_57_6_5 Unassigned
Bump_81_0_8 Assigned scan_in_1 IOPADS_INST/Pscanin1ip
Bump_86_5_8 Assigned port_pad_data_in[0] IOPADS_INST/Ptdspip00
Bump_21_0_2 Assigned port_pad_data_in[1] IOPADS_INST/Ptdspip01
Bump_8_7_0 Assigned tdigit[4] IOPADS_INST/Ptdigop4
Bump_18_7_1 Assigned tdigit[5] IOPADS_INST/Ptdigop5
Bump_45_4_4 Assigned tdigit[7] IOPADS_INST/Ptdigop7
Bump_40_9_3 Unassigned
Bump_38_7_3 Unassigned
Bump_49_8_4 Unassigned
Bump_87_6_8 Unassigned
Bump_56_5_5 Unassigned
Bump_53_2_5 Assigned port_pad_data_in[2] IOPADS_INST/Ptdspip02
Bump_20_9_1 Assigned tdigit_flag IOPADS_INST/Ptdigfgop
Bump_28_7_2 Assigned port_pad_data_out[0] IOPADS_INST/Ptdspop00
Bump_15_4_1 Unassigned
Bump_9_8_0 Unassigned
Bump_60_9_5 Unassigned
Bump_58_7_5 Unassigned
Bump_69_8_6 Unassigned
Bump_54_3_5 Unassigned
Bump_65_4_6 Unassigned
Bump_59_8_5 Assigned port_pad_data_out[2] IOPADS_INST/Ptdspop02
Bump_39_8_3 Assigned port_pad_data_out[3] IOPADS_INST/Ptdspop03
Bump_70_9_6 Assigned port_pad_data_out[1] IOPADS_INST/Ptdspop01
Bump_77_6_7 Unassigned
Bump_50_9_4 Unassigned
Bump_68_7_6 Unassigned
Bump_36_5_3 Unassigned
Bump_80_9_7 Unassigned
Bump_90_9_8 Unassigned
Bumps with signal assigned: 57 (with bump_connect_target: 0);
```

```
Bumps with power assigned: 0 (with bump_connect_target: 0);
Bumps with ground assigned: 0 (with bump_connect_target: 0);
Bumps unassigned: 33;
Total bumps: 90;
Signal/PG ratio: NA;
Min center-to-center bump pitch: 100.000
*** End the statistics information for bumps ***


*** Starting Check Bump Overlap ...
*** End Check Bump Overlap ***
Bumps overlap: 0
```

- The following command checks whether bumps are on bump grids:

  ```
  check_bumps -on_bump_grid -out_file bump.rpt
  ```

  When `-on_bump_grid` is specified, other checks are not started. Therefore, the *Bump On Bump Grid* section is the only section in the report file:

  ```
  #...header...#
  *** Starting Bump On Bump Grid ...
  'Bump_295' is off bump grid "Grid_A".
  'Bump_296' is off bump grid "Grid_B".
  'Bump_297' needs to be checked on multiple conflicted bump grids: "Grid_A" on "Layer10" and "Grid_B" on "Layer11".
  'Bump_298' won't be checked. No bump grid on the bump layer: Layer8, Layer9.
  *** End Check Bump On Bump Grid ***
  Bumps off Bump Grid: 2
  Bumps on Multiple Conflicted Bump Grids: 1
  Bumps with No Bump Grid:1
  ```

Related Information

- assign_bumps
- create_bump
- create_bump_grid

check_flip_chip_constraints

```
check_flip_chip_constraints
[-help]
[-route_style {manhattan | 45_degree_route}]
```

Reports the status of bump placement and routing contsraints. Use this command to check whether the flip chip routing result meets the settings in the constraints file.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `check_flip_chip_constraints` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man check_flip_chip_constraints`.
`-route_style {manhattan | 45_degree_route}`	Specifies the route style to check, which could be `manhattan` or `45_degree_route`. *Data_type*: enum, optional

Example

- The following command checks for flip chip routing violations of the Manhattan route style:
  ```
  check_flip_chip_constraints -route_style manhattan
  ```

Related Information

- `route_flip_chip`
- Attributes in the `flip_chip` category
- Flip Chip Methodologies chapter of the *User Guide*
- Tools Menu chapter of the *Menu Reference*
  - *Flip Chip*

check_io_inst_overlap

```
check_io_inst_overlap [-help]
```

Checks to see if any area I/O instance is overlapped with another area I/O instance. If an overlap occurs, a warning message is dumped to the console window, and the overlapped instances are highlighted in the work area.

Parameters

`-help`	Prints a brief description of the `check_io_inst_overlap` command. For a detailed description of the command, use the `man` command: `man check_io_inst_overlap`

copy_bumps

```
copy_bumps
[-help]
[-assignment]
[-name_format {format}]
-shift {X Y}
{-bumps {bump_list} | -selected }
```

Copies bumps to the specified location with the same pitch constraint. By default, the target bumps after copy are unassigned and named with the default format `Bump_%i` by the tool. Here, `%i` refers to the index of the bump, which starts at one more than the number of existing bumps in the design. Use the option `-name_format` to name the copied bumps. If there are any name collisions, an error is given and the bumps are not copied.

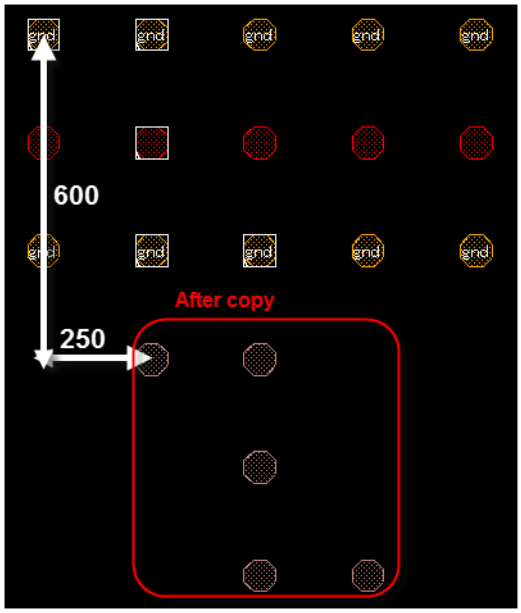
Parameters

`-help`	Prints a brief description that includes type and default information for each `copy_bumps` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man copy_bumps`
`-assignment`	Copies the bumps and their assigned signals. *Data_type*: bool, optional
`-bumps {bump_list}`	Specifies names of the bumps to be copied. You can use wildcards in the bump list. *Data_type*: string, optional
`-name_format {format}`	
	Specifies the name format for the target bumps. The format is similar to the `printf` format, except that it uses `%i` for index and `%o` for original name of source bump. `%i` mean the index of bump, which starts at one more than the number of existing bumps in the design. *Data_type*: string, optional
`-selected`	Copies only the selected bumps. *Data_type*: bool, optional
`-shift {X Y}`	Specifies the distance in microns from the source location to the target location in X/Y direction. *Data_type*: string, required

Examples

- The following command copies the selected bumps, `Bump_1` to `Bump_5`, and pastes them with a `{250 -600}` shift. The copied bumps are named `Bump_1_copy`, `Bump_2_copy`, and so on.

  ```
  copy_bumps -selected -shift {250 -600}  -name_format {%o_copy}
  ```



- Assuming that the number of bumps in the design is 15, the following command copies the selected bumps, `Bump_1` to `Bump_5`, and pastes them with a {250 -600} shift. The copied bumps are named `Bump_1_copy_16`, `Bump_2_copy_17`, …, and `Bump_5_copy_20`:

  ```
  copy_bumps -selected -shift {250 -600} -name_format {%o_copy_%i}
  ```

- Assuming that the number of bumps in the design is `15`, the following command copies the selected bumps, `Bump_1` to `Bump_5`, and pastes them with a `{250 -600}` shift. The copied bumps are named `Bump_16`, `Bump_17`, …, and `Bump_20`, according to the default name format:

  ```
  copy_bumps -selected -shift {250 -600}
  ```

Related Information

- `create_bump`
- `update_bump_name`
- `delete_bumps`
- Flip Chip Methodologies chapter

create_bump

```
create_bump
[-help]
[-allow_outside_of_die]
[-allow_overlap_control {keep_all keep_existing_bumps keep_new_bumps}]
-cell bumpcell_name
[-check_drc]
[-edge_spacing {left bottom right top}]
[-exclude_bump_area {rect_list}]
[ [-keep_out_region width height] | [-keep_out_pitch value] ]
[-location x y]
[-location_type {cell_center | cell_lower_left | geometry_center | geometry_lower_left}]
[-name_format {string, %i, %r, %c}]
[-orient {r0 r90 r180 r270 mx mx90 my my90}]
[-respect_placement_blockage]
[-return_bumps_name]
[-start_index index]
[-pitch x y [-pattern_full_chip | -pattern_side {side width} | -pattern_array {row column} | -pattern_ring width | -pattern_center {row
column}]]
[-stagger_type type [-stagger_offset value]]
[-relative_type {embedded_bump inst_pin_port block}
  [[-relative_objs {inst_pin_port_list}]
   [-relative_block inst+] [-relative_block_constraint fileName]
   [-relative_assignment] [-relative_offset x y]]]
[{-pattern_full_chip  | -pattern_array {row column} | -pattern_center {row column}} [-start_from <type>]]
```

Creates bumps and instances of bump cells based on the specified pattern. Use one of the following parameters to specify the bump pattern in the chip:

- `-pattern_full_chip`
- `-pattern_side {side width}`
- `-pattern_array {row column}`
- `-pattern_ring width`
- `-pattern_center {row column}`

If none of the above patterns is specified, `create_bump` creates a single bump at the specified location.

Bumps are usually placed in the top design. However, some blocks may also include bumps. Such bumps are called embedded bumps. When implementing the block design, use the following special keywords in the LEF definition for embedded bumps.

- Add following property keywords in the head of tech LEF file

  ```
  PROPERTYDEFINITION
      LAYER LEF58_TYPE STRING ;
  END PROPERTYDEFINITIONS
  ```

- Add PASSIV LAYER definition after AP(top) layer

  ```
  LAYER PASSIV
    TYPE CUT ;
    PROPERTY LEF58_TYPE "TYPE PASSIVATION ; " ;
  END PASSIV
  ```

- Add a same geometry on PASSIV layers with the bump pins

  ```
  MACRO BUMP
  CLASS COVER BUMP ;
    …
      PIN A
    …
        PORT
          LAYER PASSIV ;
          POLYGON 27.985 0 67.565 0 95.55 27.985 95.55 67.565 67.565 95.55 27.985 95.55 0 67.565 0 27.985 ;
          LAYER AP ;
          POLYGON 27.985 0 67.565 0 95.55 27.985 95.55 67.565 67.565 95.55 27.985 95.55 0 67.565 0 27.985 ;
        END
  ```

```
        END A
      END BUMP
```

After the block design is finished, the bumps are dumped out as one port of the pin with PASSIV layer and top layer (AP) as similar as below, tool could check if the port is the embedded bump by PASSIV layer from the top design.

```
PIN DQ0
    DIRECTION INOUT ;
    USE SIGNAL ;
    PORT
      LAYER AP ;
      POLYGON 759.9810 366.7830 799.5610 366.7830
              827.5460 338.7980 827.5460 299.2180
              799.5610 271.2330 759.9810 271.2330
              731.9960 299.2180 731.9960 338.7980 ;
      LAYER PASSIV ;
      POLYGON 759.9810 366.7830 799.5610 366.7830
              827.5460 338.7980 827.5460 299.2180
              799.5610 271.2330 759.9810 271.2330
              731.9960 299.2180 731.9960 338.7980 ;
END
    PORT
…
END
…
END DQ0
```

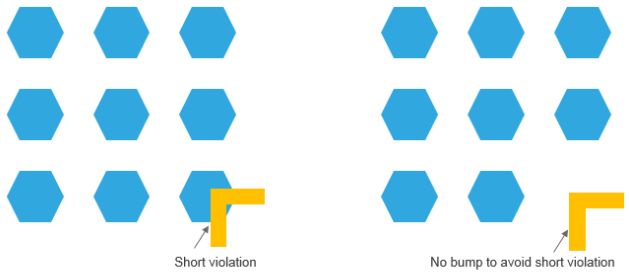
If you are creating an embedded bump, `create_bump` automatically creates a bump in the top design with the correct net assignment at the same location as the embedded bump.

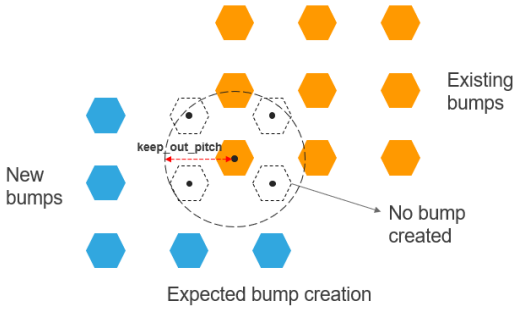
`create_bump` also enables you to create bumps based on pin location.

Note - During bump placement trials, if you click the *Undo* button (or use the `gui_undo` command) after running `create_bump`, the bump floorplan will return to the state it was in before `create_bump`. If you then click the *Redo* button (or use the `gui_redo` command), the bump floorplan will reapply the changes made by `create_bump`.

Parameters

`-help`	Prints a brief description that includes type and default information for each `create_bump` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man create_bump`
`-allow_outside_of_die`	Allows bumps to be created outside of die area. When this option is specified, the tool ignores floorplan constraints when creating a bump array pattern. This option can be used only if `-pattern_array` and `-location` are also specified. *Data_type*: bool, optional
`-allow_overlap_control {keep_all keep_existing_bumps keep_new_bumps}`	

	Specifies the overlap control type. The allowed values for the `-allow_overlap_control` parameter are: • `keep_all` - Keeps all overlapping bumps. • `keep_existing_bumps` - Keeps the existing bumps before calling `create_bump` if there is an overlap. If this option is specified,`create_bump` skips creating a bump if there is an overlap and continues creating the remaining bumps. A warning message is displayed when bump creation is skipped to avoid overlap. • `keep_new_bumps` - Removes the existing bumps before calling `create_bump`, places the newly created bump if there is overlap, and then continues creating the remaining bumps. A warning message is displayed when deleting the existing bumps to avoid overlap. If the status of the existing bumps to be deleted is covered or fixed, a warning is displayed and the new bumps overlapping with the covered or fixed old bumps are not kept. ⚠ `create_bump`: • Checks overlaps between regular bumps by checking for pin overlaps on bumps. • Checks overlaps between dummy bumps by checking for OBS overlap. • Checks overlaps between dummy bumps and regular bumps by checking for OBS and bump pin overlap. *Data_type*: enum, optional
`-cell` *bumpcell_name*	Specifies the name of the bump cell, which is used to create bumps. *Data_type*: string, required
`-check_drc`	Checks DRC rules and does not create a bump at any location where there is a violation. A suitable warning message is also issued. For example, as indicated in the following diagram, no bump is created at the location where a short violation is detected when `-check_drc` is specified with `create_bump`.  Bump creation without -check_drc Bump creation with -check_drc In addition, a warning such as the following is displayed: `**WARN: (IMPSIP-9068): New bump at (0.000, 75.000) has short violation and skipped.` Currently, `create_bump -check_drc` checks for the following types of violations: • Short violation • Min spacing violation • Cut spacing violation • EOL (end of line) violation *Data_type*: bool, optional
`-edge_spacing {`*left bottom right top*`}`	
	Specifies the minimum distances in microns to the edge of the chip from the lower left or center of the outermost bumps. The `-location_type` value determines which point of the bump (lower-left or center of the cell or geometry) is used to compute the distance to the edge. The arrays start at the left, bottom location and extend to the right and top. *Data_type*: string, optional
`-exclude_bump_area {`*rect_list*`}`	

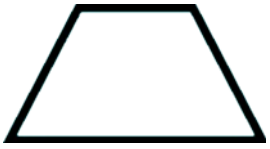
	Specifies the area to be excluded from bump placement. No bump geometry is allowed inside the specified area. It supports multiple rectangles, such as `-exclude_bump_area {llx1 lly1 urx1 ury1 llx2 lly2 urx2 ury2 ...}`. *Data_type*: string, optional
`-keep_out_pitch value`	Defines a circular keep-out area with the bump center as the center and the specified pitch value as the radius. By default, the tool will not create a bump at a location if an existing bump violates the center-to-center pitch at that location, as shown below.  If the bump-center-to-bump-center pitch at a location is smaller than the specified keep-out pitch value, the tool displays a warning. Whether the tool then retains the existing bumps in the overlap region or replaces them with the new bumps depends on the `-allow_overlap_control` parameter setting: • If `-keep_out_pitch` is specified but `-allow_overlap_control` is not specified, the tool keeps the existing bumps and does not add new bumps in the overlap region. This is the same behavior as when `-keep_out_pitch` is specified along with `-allow_overlap_control keep_existing_bumps`. • If `-keep_out_pitch` is specified along with `-allow_overlap_control keep_new_bumps`, the tool keeps the newly added bumps and deletes the existing bumps. If the bump-center-to-bump-center pitch equals the specified keep-out pitch value, the tool creates bumps without any violations. *Data_type*: float, optional
`-keep_out_region width height`	Defines a dynamic, keep-out region where bumps are not going to be created if there are existing bumps in that region. The keep-out region is a box area of the specified width and height, and its center is the same as the bump center. By default, if you specify `-keep_out_region` and there are existing bumps in the region, the tool displays a warning and keeps the existing bumps in the overlap area. Specify `-allow_overlap_control keep_new_bumps` if you want to keep the new bumps and delete existing ones in the case of overlap in the specified keep-out region. *Data_type*: point, optional
`-location x y`	Specifies the `x` and `y` coordinates of the start point for bump creation based on location type. The unit is micron. For different bump patterns, the `-location` coordinates have different meanings for the start point: • For `-pattern_full_chip`, `-pattern_side {bottom width}`, `-pattern_side {left width}`, `-pattern_array {row column}`, `-pattern_ring width`, the start point is the location of the lower-left bump in the bump pattern. • For `-pattern_side {top width}`, the start point is the location of the upper-left bump in the bump pattern. • For `-pattern_side {right width}`, the start point is the location of the lower-right bump in the bump pattern. • For `-pattern_center {row column}`, the start point is the center of the bump pattern. If the location violates the spacing rule defined by the option `-edge_spacing`, the tool issues a warning but still uses the `-location` value for the start point. *Data_type*: point, optional
`-location_type {cell_center | cell_lower_left | geometry_center | geometry_lower_left}`	

	Specifies the location type for bump creation. The default type is `cell_center`. *Data_type*: enum, optional
`-name_format {`*`string, %i, %r, %c`*`}`	
	Specifies the format for the bump name like a `printf` format, except that it uses `%i` for index, `%r` for row number, and `%c` for column number. The index starts at one more than the number of existing bumps in the design, unless `-start_index` is specified and increments for each bump created. This means if there are no bumps in the design, the index starts at `1` and if there are 100 bumps in the design, the index starts at 101. The default name format is `Bump_%i`. Therefore, if there are no bumps in the design, the tool generates bumps named `Bump_1`, `Bump_2`, and so on, by default. Some examples of the possible format are: • `-name_format` *`prefix_%i_string_%r_%c_suffix`* • `-name_format %i` • `-name_format` *`bumpname`* If there are any name collisions, an ERROR occurs and the bump pattern is not created. *Data_type*: string, optional
`-orient {r0 r90 r180 r270 mx mx90 my my90}`	
	Specifies the orientation of the bump for bump generation. Use this option if you are creating a pillar bump. A pillar bump can be rectangular and you may rotate the pillar bump by 90/180/270 degrees to alleviate package and chip routing problem. *Default:* `r0` **Note**: If you are creating a regular C4 bump, which is either octagonal or square in shape, rotation does not change its footprint in silicon. *Data_type*: enum, optional
`-pattern_array {`*`row column`*`}`	Creates an array of bumps. The bump starts at the coordinates specified with `-location`, and extends to the number of *`row`* and *`column`* values given from the start corner specified using `-start_from`. *Data_type*: string, optional
`-pattern_center {`*`row column`*`}`	Creates an array of bumps that is centered on the center of the chip and that extends for the *`row`* and *`column`* values given. The center point can optionally be set directly with `-location`. The starting corner of the array can be specified using the `-start_from` parameter. *Data_type*: string, optional
`-pattern_full_chip`	Creates an array of bumps covering the chip. The array starts at the closest to boundary point set by `-edge_spacing` at the corner specified by `-start_from` and extends until within `-edge_spacing` of the chip boundary for the opposite sides. You can optionally directly specify the `-location` value for the starting point, in which case `-edge_spacing` affects only the two opposite edges of the array. For examples, see the Examples section below. *Data_type*: bool, optional
`-pattern_ring` *`width`*	Creates a ring of bumps around the boundary of the chip that is *`width`* bumps wide, where *`width`* is an integer value. The ring extends until within `-edge_spacing` of the chip boundary, starting at the lower-left value allowed by `-edge_spacing`. You can optionally directly specify the `-location` value for the lower-left bump location, in which case `-edge_spacing` only affects the right and top edges of the array. The `-pitch` *`x`* value is the spacing between bumps inside one ring, and the `-pitch y` value is the spacing between two rings. For examples, see the Examples section below. *Data_type*: int, optional

`-pattern_side {side width}`	Creates an array of bumps along one side of the chip. The array is the length of the side, and *`width`* bumps wide, where *`width`* is an integer value. The legal *`side`* names are: `left`, `top`, `right`, `bottom`. The array starts at the left/bottom controlled by `-edge_spacing`, and extends until within `-edge_spacing` of the top/right sides. You can optionally directly specify the `-location` value for the lower-left bump location for bottom/left sides, the upper-left bump for the top side, or lower-right for the right side. For examples, see the Examples section below. *Data_type*: string, optional
`-pitch x y`	Specifies the distance between bump centers in microns. `-pitch {x y}` is interpreted in different ways for different bump patterns. • For `full_chip`, bottom_side row, top_side row, left_side column, right_side column, array {row column}, and center {row column}: ◦ `-pitch x` means the distance between the bump centers in a row or horizontal. ◦ `-pitch y` means the distance between the bump centers in column or vertical. • For `ring ring_number`: ◦ `-pitch x` means the distance between the bump centers in the ring. ◦ `-pitch y` means the distance between rings from the bump center. *Data_type*: point, optional
`-relative_assignment`	Assigns the corresponding net of the relative object to the bump being created. This option can be used only if `-relative_type` is specified. If `relative_assignment` is specified and `-relative_type` is: • `embedded_bump`: `create_bump` searches the top netlist for the net connected to the pin of the embedded bump and assigns it to the created bump. • `inst_pin_port`: You must also specify `-relative_objs {inst_pin_port_list}`. `create_bump` then searches the top netlist for the net connected to the specified pin and assigns it to the created bump. • `block`: You must also specify `-relative_block {inst+}`. `create_bump` then searches the top netlist for the net connected to the top layer pins of the specified blocks and assigns it to the created bump. **Note**: This option is used with `-cell`, `-start_index`, `-name_format`, `-orient`, -allow_overlap_control , `-relative_type`, `-relative_objs` and `-relative_offset`. An error message is displayed if it is used with any other option. *Data_type*: bool, optional
`-relative_block inst+`	Specifies the block list for creating top-level bumps based on block pins on the top layer. The `-relative_block` parameter must be specified when `-relative_type` is set to `block`. The `-relative_block` parameter supports wildcards. For example, you can specify `-relative_block *`. *Data_type*: (inst)+, optional
`-relative_block_constraint fileName`	

	Specifies the filename of the block constraint to be used for creating top-level bumps based on block pins on the top layer. The `-relative_block_constraint` parameter is optional and can be used only when `-relative_type block` is specified.
	
	`create_bump` parses the constraint file specified with `-relative_block_constraint` and creates bumps based on the instance pins specified with `-relative_block`. The top-level bump is created at the center of the block pin geometries. Note that bumps are created on a block pin only if it is on the top layer of a design.
	
	The constraint format for bumps based on a block pin is as follows:
	
	`BUMP_ON_BLOCK_PIN`
	`  MACRO` *`marco_name_list`*
	`  PIN [`*`pin_name_list`*`] [SINGLE|MULT]`
	`  PORT [SINGLE|MULT]`
	`  GEOMETRY_SHAPE RECT/POLY`
	`  GEOMETRY_SHORT_EDGE` *`min_value`* `[:`*`max_value`*`]`
	`  GEOMETRY_LONG_EDGE` *`min_value`* `[:`*`max_value`*`]`
	`  NET` *`net_list_name`*
	`END BUMP_ON_BLOCK_PIN`
	
	Note: If a constraint file is specified, the tool creates bumps only on the block pins that are specified in the constraint file. If you do not specify a constraint file, the tool creates a single bump for each block pin on the top layer for the specified blocks. If a constraint file is specified but is empty, the tool does not create any bumps because no block pin is specified in the constraint file.
	
	To see the use model and examples of the `BUMP_ON_BLOCK_PIN` constraint, refer to the "Bump Creation" section in the Flip Chip Methodologies chapter in the *User Guide*.
	
	Data_type: string, optional

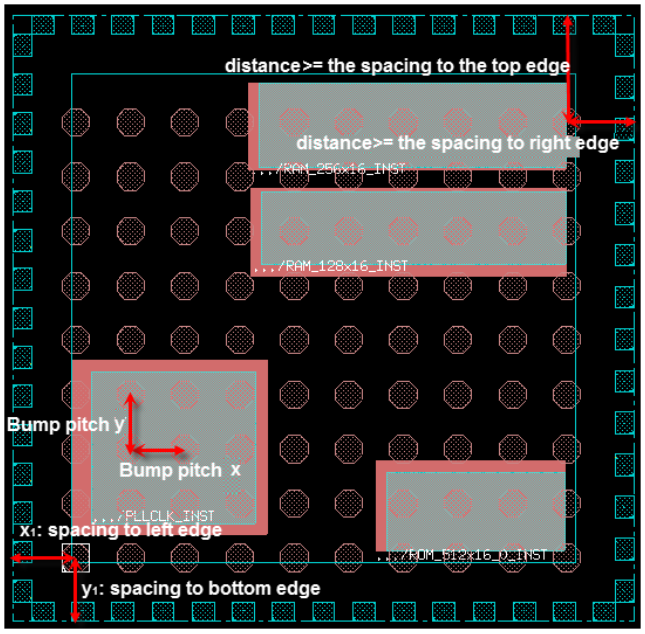
`-relative_objs {`*`inst_pin_port_list`*`}`	
	Specifies the inst_pin_port list and supports wild cards. The format is `{inst:pin:port ...}` and `port` is of type integer. When the `-relative_objs` option is specified, bumps are created by default at the geometry center of the specified ports. The bump is named *`$blockName-$pinName-Port_$portNumber-embeddedBump_$index`*, where: • *`$blockName`* is the name of the block in the top design which has the corresponding embedded bump. • *`$pinName`* is the name of the pin in LEF which has the corresponding embedded bump. • *`$portNumber`* is the number of the port on which the created bump is based. **Example**: `Avd_test_1-DQ0-Port_1-embeddedBump_1` You can reset the default settings by using the `-start_index` and `-name_format` options. **Note**: This option is used with `-cell`, `-start_index`, `-name_format`, `-orient`, -allow_overlap_control-relative_type, -relative_assignment and -relative_offset. An error message is displayed if it is used with any other option. *Data_type*: string, optional
`-relative_offset` *`x y`*	Specifies the offset of bump location compared with the relative object. The geometry center of the created bump is the geometry center of the relative object plus the offset. This option can be used only if `-relative_type` is specified. **Note**: This option is used with `-cell`, `-start_index`, `-name_format`, `-orient`, -allow_overlap_control, `-relative_objs` and `-relative_assignment`. An error message is displayed if it is used with any other option. *Data_type*: point, optional
`-relative_type {embedded_bump inst_pin_port block}`	

	Specifies the type of the relative object: • `embedded_bump`: When the `embedded_bump` type is specified, `create_bump` automatically creates a bump at the same location as the embedded bump. The bump is named *`$blockName/$pinName/embeddedBump_$index`*, where: ◦ *`$blockName`* is the name of the block in the top design that has the corresponding embedded bump. ◦ *`$pinName`* is the name of the pin in LEF that has the corresponding embedded bump. You can reset the default settings by using the `-start_index` and `-name_format` options. If there are already overlapping bumps for some embedded bumps, `create_bump` skips creating bumps for those embedded bumps but continues to create bumps for remaining embedded bumps and issues the following warning message. `WARNING (#): There is(are) overlap bump(s) with the embedded bump $blockName/$pinName located at {x2 y2} with relative offset {x y}. Skip creating bump for it.` After all bumps are created, `checkBump` displays a summary of the number of embedded bumps for which the tool failed to create bumps. • `inst_pin_port`: When the `inst_pin_port` type is specified, `create_bump` creates bumps as per the *`inst_pin_port_list`* you specify with `-relative_objs`. Therefore, if you specify `-relative_type` as `inst_pin_port`, you must specify the `-relative_objs {inst_pin_port_list}` option. • `block`: When the `block` type is specified, `create_bump` automatically creates bumps at the same location as that of the specified block pin, which must be on the top routing layer of the design. The `-relative_block` parameter must be specified when `-relative_type` is set to `block`. *Data_type*: enum, optional
`-respect_placement_blockage`	Specifies that the bumps being created, including dummy bumps, should not overlap any placement blockages. *Data_type*: bool, optional
`-return_bumps_name`	Returns the names of the bumps generated. If `create_bump` fails to generate any bumps, the tool returns an empty string. *Data_type*: bool, optional
`-stagger_offset` *value*	Specifies the offset between different rows or columns in microns. This option only works with `-pattern_side`. *Data_type*: float, optional
`-stagger_type {odd_rows odd_columns even_rows even_columns trapezoid }`	
	Specifies the stagger type pattern. The `-stagger_type` parameter cannot be used with `-pattern_ring`. You can use the `-stagger_offset` parameter along with `-stagger_type` to specify the offset of bumps between rows or columns in the generated bump pattern. The allowed values for the `-stagger_type` parameter are: • `odd_rows` - Creates a bump array with the odd rows shifted to the right by the specified stagger offset value. • `odd_columns` - Creates a bump array with the odd columns shifted up by the specified stagger offset value. • `even_rows` - Creates a bump array with the even rows shifted to the right by the specified stagger offset value. • `even_columns` - Creates a bump array with the even columns shifted up by the specified stagger offset value. • `trapezoid` - Creates bumps are created in a trapezium pattern. This stagger type is supported only with `-pattern_side`.  For examples of how `-stagger_offset` affects different `-stagger_type` patterns, refer to the Examples section below. *Data_type*: enum, optional
`-start_from {lower_left upper_left lower_right upper_right}`	

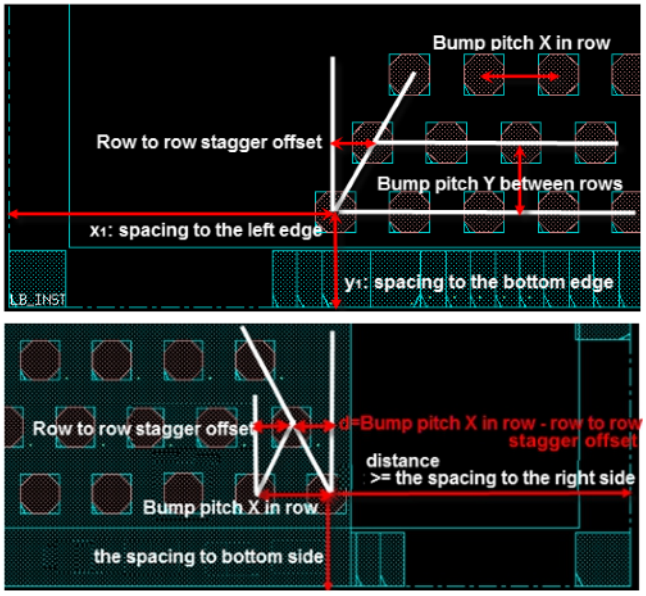
	Specifies the starting corner for creating an array of bumps. This option can be used only if `-pattern_array`, `-pattern_center`, or `-pattern_full_chip` is specified. **Note**: Bumps are always indexed by row first, irrespective of the `-start_from` value. For examples of using `-start_from`, refer to the Examples section below. *Default*: `lower_left` *Data_type*: enum, optional
`-start_index` *index*	Specifies the starting number of the index for bumps to be created. *Data_type*: int, optional

Examples

- The following command creates a single bump at the lower-left corner of the bump cell box:
  ```
  create_bump -cell BUMPCELL -location_type cell_lower_left -location {0 0}
  ```
- The following command uses the default location type setting and creates an array of bumps with bump cell that has the same size as the bump bounding box for the macro BUMPCELL:
  ```
  create_bump -cell BUMPCELL -pitch {x y} -edge_spacing {x1 y1 x2 y2} -pattern_full_chip
  ```



- The following command creates a full bump matrix on the chip with the spacing constraint {`400 500 450 600`} and bump pitch {`80 120`}. The names of the created bumps honor the default naming format `Bump_%i`, where `%i` means the index:
  ```
  create_bump -cell BUMPCELL -pitch {80 120} -edge_spacing {400 500 450 600} -pattern_full_chip
  ```
- The following command creates a 3-row array of bumps along the bottom of the chip:
  ```
  create_bump -cell BUMPCELL -pitch {X Y} -edge_spacing {x1 y1 x2 0} -pattern_side {bottom 3} -stagger_offset X
  ```

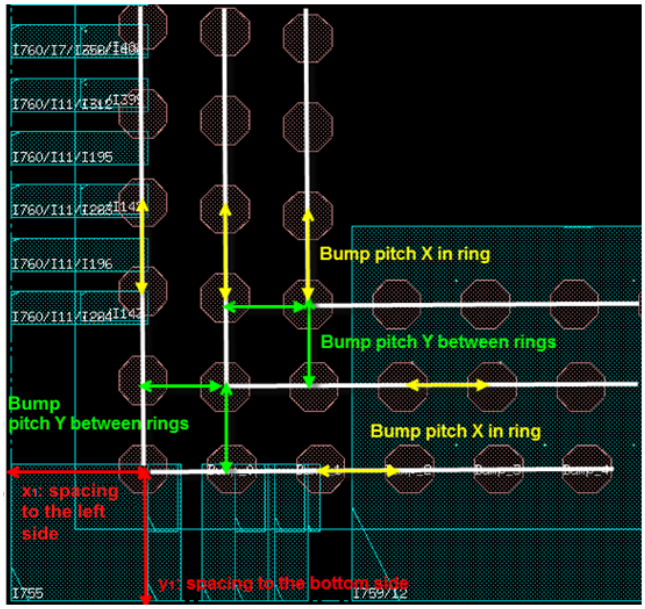


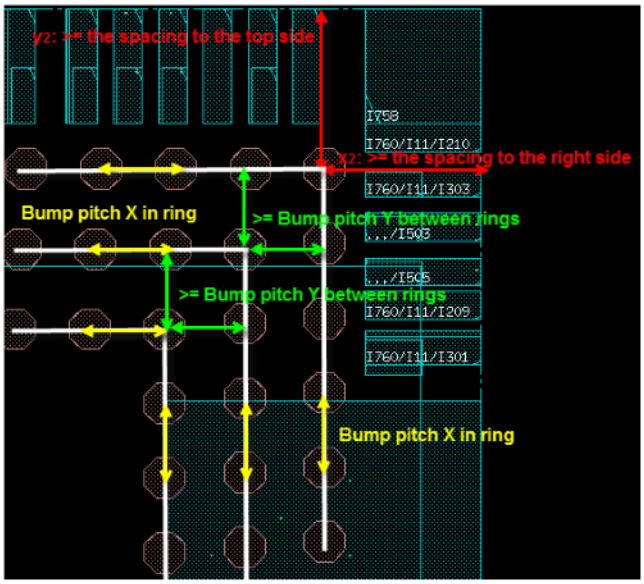
- The following command creates a 3-row bump pattern on the bottom side with stagger offset `50` and bump pitch {`80 120`}. The names of the created bumps honor the format of `bottom_%i`, where `%i` means the index:

  ```
  create_bump -name_format bottom_%i -cell BUMPCELL -pitch {80 120} -edge_spacing {400 400 500 0} -pattern_side {bottom 3} -
  stagger_offset 50
  ```

- The following command creates a ring of bumps around the boundary of the chip that is 3 bumps wide:

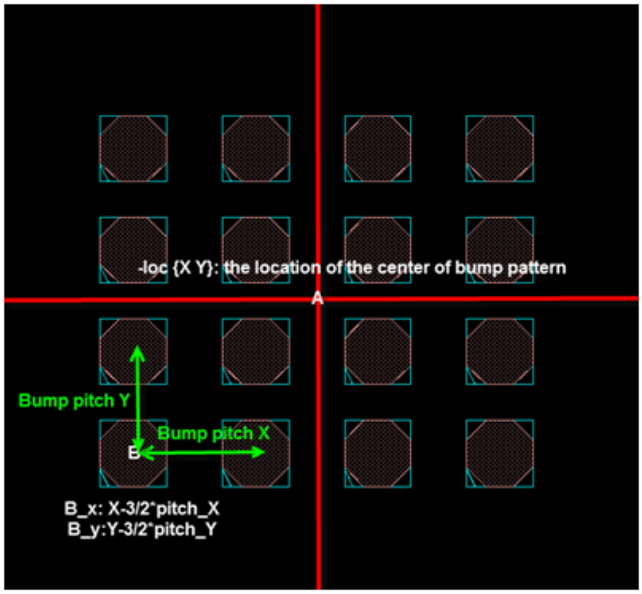
  ```
  create_bump -cell BUMPCELL -pitch {X Y} -edge_spacing {x1 y1 x2 y2} -pattern_ring 3
  ```





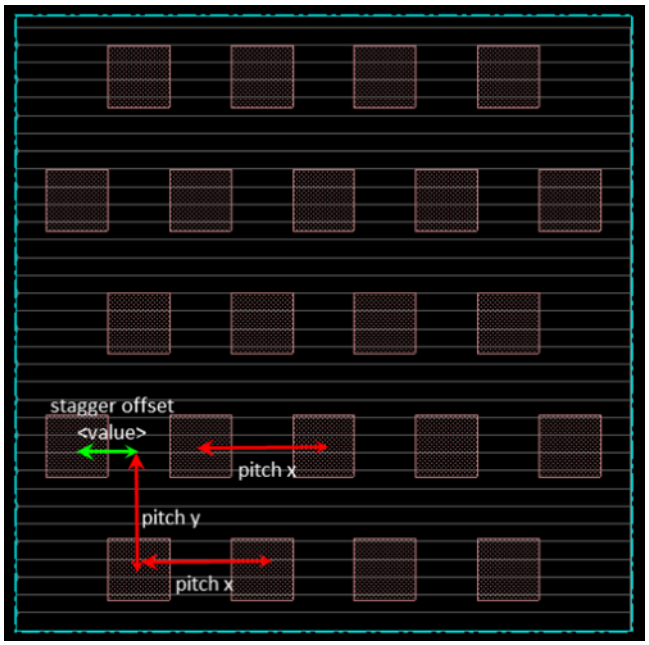
- The following command creates a 4x4 array of bumps that is centered on the point defined by `-location`:

  `create_bump -cell BUMPCELL -pitch {X Y} -location {X Y} -pattern_center {4 4}`

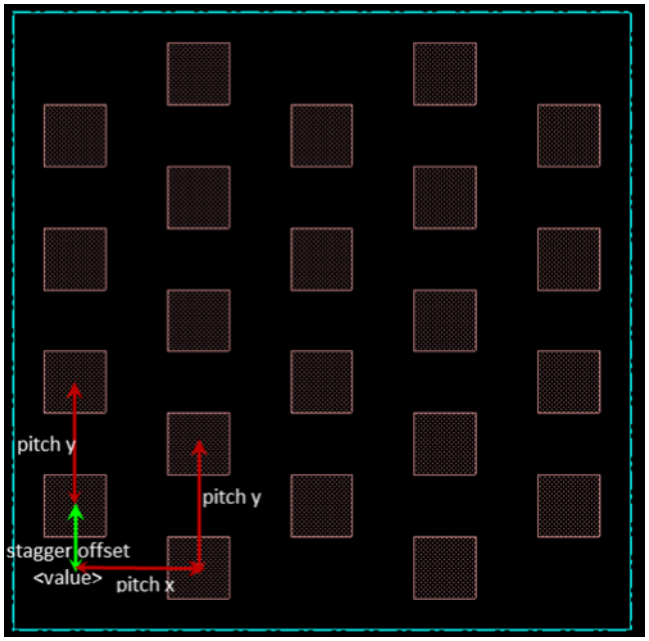


- The following example shows how `-stagger_offset` can be used to create a 5X5 array of bumps in `-stagger_type odd_rows` stagger pattern:

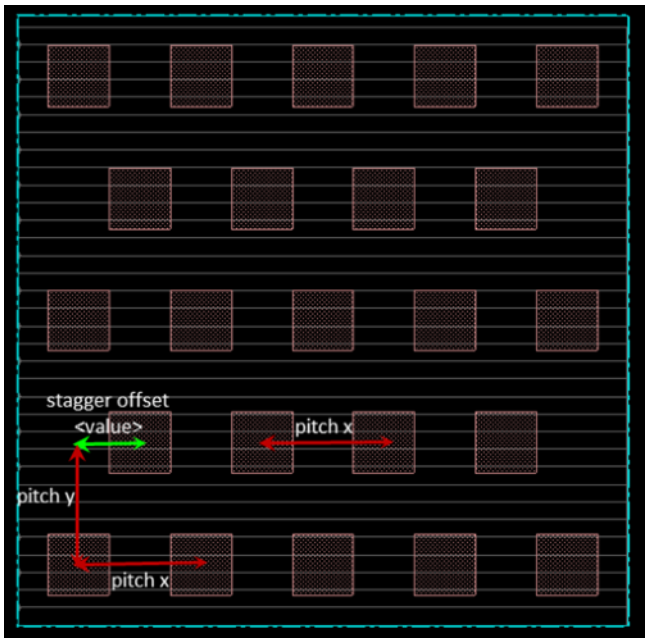
  `create_bump -cell BUMPCELL -location {loc_x loc_y} -pitch {pitch_x pitch_y} -pattern_array {5 5} -stagger_type odd_rows -stagger_offset value`



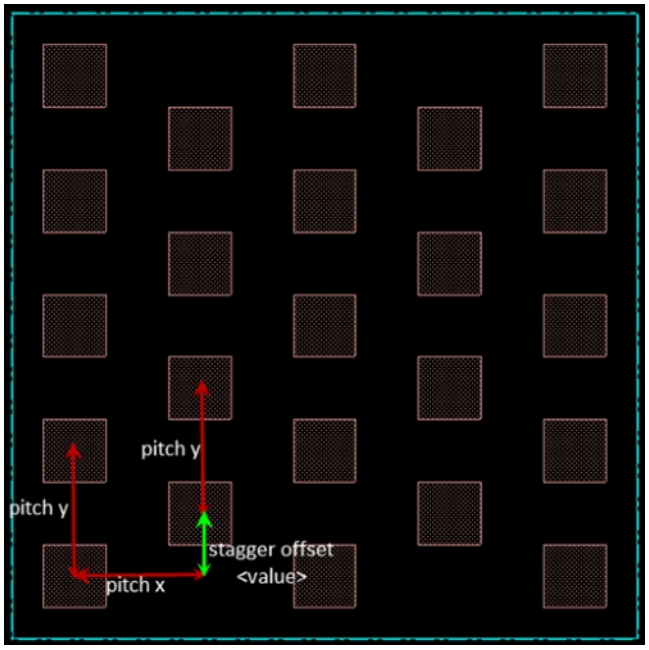
- The following example shows how `-stagger_offset` can be used to create a 5X5 array of bumps in `-stagger_type odd_columns` stagger pattern:
  `create_bump -cell BUMPCELL -location {loc_x loc_y} -pitch {pitch_x pitch_y} -pattern_array {5 5} -stagger_type odd_columns -stagger_offset value`



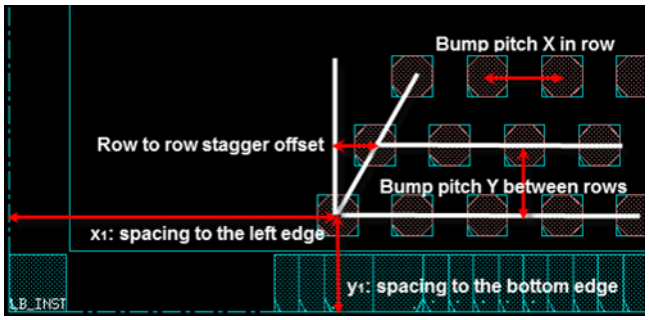
- The following example shows how `-stagger_offset` can be used to create a 5X5 array of bumps in `-stagger_type even_rows` stagger pattern:
  `create_bump -cell BUMPCELL -location {loc_x loc_y} -pitch {pitch_x pitch_y} -pattern_array {5 5} -stagger_type even_rows -stagger_offset value`



- The following example shows how `-stagger_offset` can be used to create a 5X5 array of bumps in `-stagger_type even_columns` stagger pattern:
  `create_bump -cell BUMPCELL -location {loc_x loc_y} -pitch {pitch_x pitch_y} -pattern_array {5 5} -stagger_type even_columns -stagger_offset value`

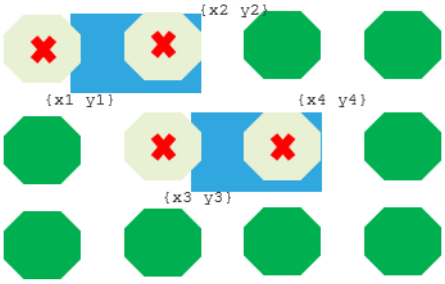


- The following example shows how `-stagger_offset` can be used with `-stagger_type trapezoid` stagger pattern to create a 3-row bump pattern on the bottom side:
  `create_bump -cell BUMPCELL -edge_spacing {left bottom right top} -pitch {pitch_x pitch_y} -pattern_side {bottom 3} -stagger_type trapezoid -stagger_offset value`



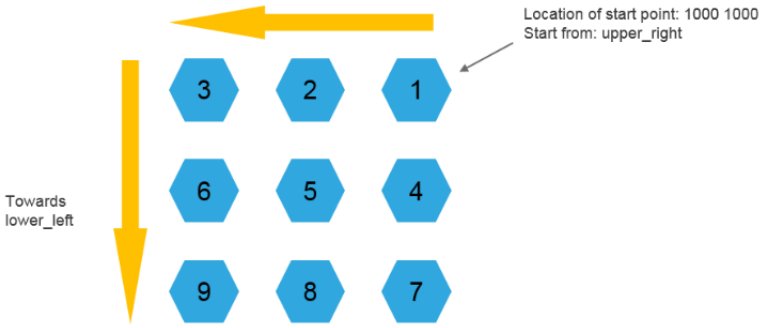
- The following command creates a 3X4 array of bumps but does not place bumps in the:
  - rectangular area with lower left corner at {x1 y1} and upper right corner at {x2 y2}
  - rectangular area with lower left corner at {x3 y3} and upper right corner at {x4 y4}

```
create_bump -cell BUMPCELL -location {x y} -pattern_array {3 4} -exclude_bump_area {x1 y1 x2 y2 x3 y3 x4 y4}
```



- The following command creates a 3X3 array of bumps with the upper-right corner of the array starting at point (1000, 1000):

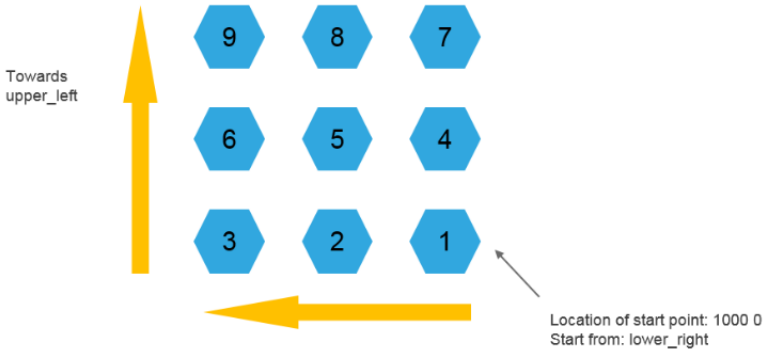
```
create_bump -pattern_array {3 3} -location 1000 1000 -cell BUMP  -pitch 200 200 -start_from upper_right
```



The array starts from the upper right and continues towards lower left.

- The following command creates a 3X3 array of bumps with the lower-right corner of the array starting at point (1000, 0):

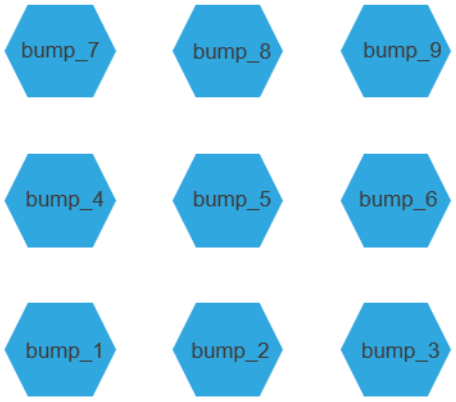
```
create_bump -pattern_array {3 3} -location 1000 0 -cell BUMP -pitch 200 200 -start_from lower_right
```



The array starts from the lower right and continues towards upper left.

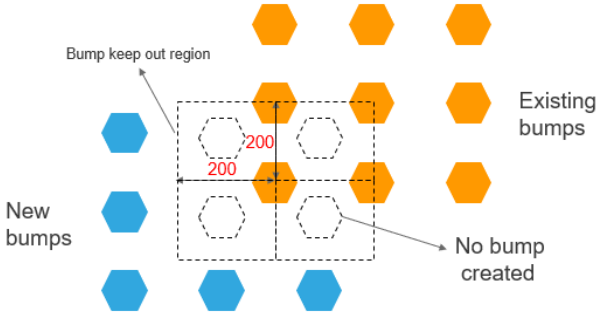
- The following command creates a 3X3 array of bumps at the specified location and returns the names of the bumps generated:

```
create_bump -pattern_array {3 3} -location 0 1000 -cell BUMP -pitch 200 200 -return_bumps_name
> bump_1 bump_2 bump_3 bump_4 bump_5 bump_6 bump_7 bump_8 bump_9
```



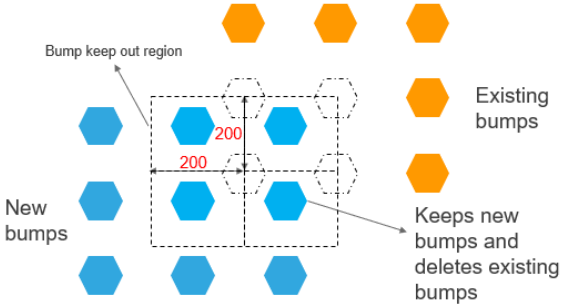
- The following command creates a 3X3 bump array and does not add new bumps in the specified keep-out region in the case of an overlap:

```
create_bump -pattern_array {3 3} -location 200 200 -cell BUMP -pitch 200 200 -keep_out_region {200 200}
```



- The following command creates a 3X3 bump array and does not add new bumps in the specified keep-out region in the case of an overlap:

```
create_bump -pattern_array {3 3} -location 200 200 -cell BUMP -pitch 200 200 \
-keep_out_region {200 200} -allow_overlap_control keep_new_bumps
```



- The following command creates a 5X5 bump array and allows bumps to be created outside of the die area:

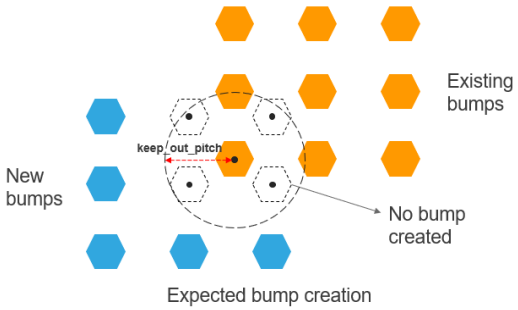
```
create_bump -cell BUMPCELL -start_from upper_right -location 0 0 -pitch 100 100  -pattern_array {5 5} -allow_outside_of_die
```

- The following command parses the `1.const` constraint file and creates bumps based on the specified instance pins:

```
create_bump -cell BUMPCELL -relative_type block -relative_block {BLOCK_LIST} -relative_block_constraint 1.const -relative_assignment
```

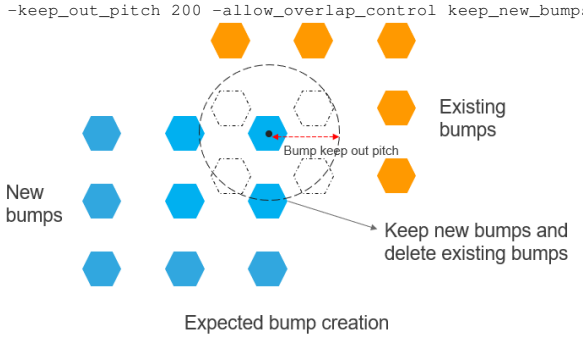
- The following command creates a 3X3 bump array and does not add new bumps in the circular area defined by the specified keep-out pitch in the case of an overlap:

```
create_bump -pattern_array {3 3} -location 200 200 -cell BUMP -pitch 200 200 -keep_out_pitch 200
```



- The following command creates a 3X3 bump array and adds new bumps in the circular area defined by the specified keep-out pitch in the case of an overlap:

```
create_bump -pattern_array {3 3} -location 200 200 -cell BUMP -pitch 200 200 \
 -keep_out_pitch 200 -allow_overlap_control keep_new_bumps
```



Related Information

- `assign_bumps`
- `copy_bumps`
- `delete_bumps`
- `gui_undo`
- `gui_redo`
- `set_bump_master`
- `unassign_bumps`
- `update_bump_name`
- Flip Chip Methodologies chapter in the *User Guide*
- Tools Menu chapter in the *Menu Reference*
  - *Flip Chip*

create_bump_connect_target_constraint

```
create_bump_connect_target_constraint
[-help]
{-bumps list_of_bumps | -selected }
{ {-io_inst instance_name
        [ {-pin_name pin_name | -net net_name}
          [-port_num value]]}
  | -pg_connect_type {ioring corering stripe iopin}}
```

Adds a string property in one or multiple bumps. If any of the specified/selected bump has a property value already, Innovus checks whether the new property value is exactly the same as the existing one. If not, the new property value is concatenated to the existing property string and separated by a space.

Using this command, you can add either a complete port numbering property (`inst:pin:port`) or an incomplete property, such as `inst` or `inst:pin` to PG bumps.

Parameters

`-help`	Prints a brief description that includes type and default information for each `create_bump_connect_target_constraint` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man create_bump_connect_target_constraint`
`-bumps` *list_of_bumps*	Adds bump connect target constraint on specified bumps. The option `-bumps` can accept wildcards. In case of multiple bumps being specified, the same property is added onto all the bumps. *Data_type*: string list, required
`-io_inst` *instance_name*	Specifies the connect target instance. If only `-io_inst` is specified and `-net` or `-pin_name` is not specified, the tool automatically finds the most suitable geometries in the instance specified by *instance_name*. *Data_type*: string, required
`-net` *net_name*	Specifies the connect target net name. **Note:** This option cannot be used with `-pin_name`. *Data_type*: string, optional
`-pg_connect_type {ioring corering stripe iopin}`	
	Specifies the PG connect type. Connection type can be I/O Ring (`ioring`), Core Ring (`corering`), Stripe (`stripe`), or I/O Pin (`iopin`). *Data_type*: enum, required
`-pin_name` *pin_name*	Specifies the connect target pin name. **Note:** This option cannot be used with `-net`. *Data_type*: string, optional
`-port_num` *value*	Specifies the connect target port number. *Data_type*: int, optional
`-selected`	Adds bump connect target constraint on selected bumps. If multiple bumps are selected, the same property is added onto all the bumps. *Data_type*: bool, required

Examples

- The following command adds `bump_connect_target` property onto `Bump_1` on `VDD` net. Then `Bump_1` pairs with `PORT1` under the PIN connecting to `VDD` net of arealo `IOPADS_INST/esd`.
  ```
  create_bump_connect_target_constraint -bumps Bump_1 -io_inst IOPADS_INST/esd -net VDD -port_num 1
  ```
- The following command adds `bump_connect_target` property onto `Bump_1`. Then `Bump_1` pairs with `PORT2` under the PIN whose name is `vdd!` of instance `DTMF_INST/PLLCLK_INST`. `PORT2` is required with PORT `CLASS BUMP` attribute in LEF.

```
create_bump_connect_target_constraint -bumps Bump_1 -io_inst DTMF_INST/PLLCLK_INST -pin_name vdd! -port_num 2
```

Related Information

- delete_bump_connect_target_constraints
- get_pin_port_number
- update_bump_connect_target_constraints
- Flip Chip Methodologies chapter of the *User Guide*
  - *Port Numbering*
- Tools Menu chapter in the *Menu Reference*
  - *Flip Chip*

create_signal_pin

```
create_signal_pin
[-help]
[-ignore_floating_pins]
{-inst inst_name_list | -pins pin_name_list}
```

Promotes signal pins in the IO pad/block to the top-level design as a physical pin for specified instances or pins. The instance's cell type should be either CLASS PAD AREAIO or CLASS BLOCK. The pins to be promoted should have geometry on the top layer. You can choose to ignore floating pins.

Parameters

`-help`	Prints a brief description that includes type and default information for each `create_signal_pin` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man create_signal_pin`
`-ignore_floating_pins`	Ignores the floating pins. If this parameter is not specified when promoting pins to the top level, the tool converts the floating pins in the instance to pgTerms because these pins are not logically connected to any terminal on the top level. *Data_type*: bool, optional
`-inst inst_name_list`	Specifies the names of instances for which pins need to be promoted. **Note**: This option cannot be used with `-pins`. *Data_type*: string list, required
`-pins pin_name_list`	Lists specific pins that are to be promoted to the top level. The specified pins should be hierarchical. **Note**: This option cannot be used with `-inst`. *Data_type*: (pin)+, optional

Examples

- The instance `lna_inst` has a signal pin called `data1_i`. The pin `data1_i` has a port geometry shape on the top layer. The lef for the `lna` cell is as follows:

```
MACRO lna

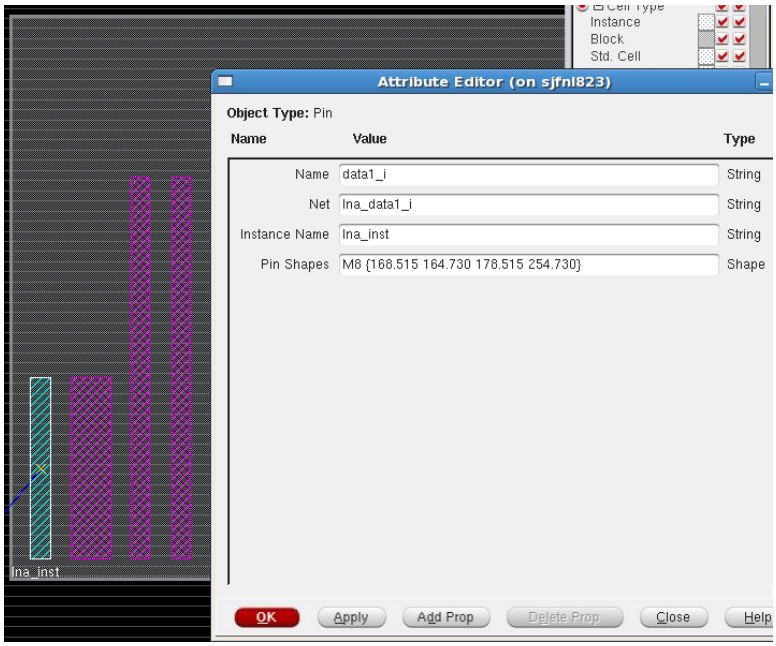
CLASS BLOCK ;

...
  PIN data1_i
    DIRECTION INPUT ;
    USE SIGNAL ;
    PORT
        CLASS BUMP ;
        LAYER METAL8 ;
        RECT   10.00 10.00   20.00 100.00 ;
    END
  END data1_i
...
END lna
```

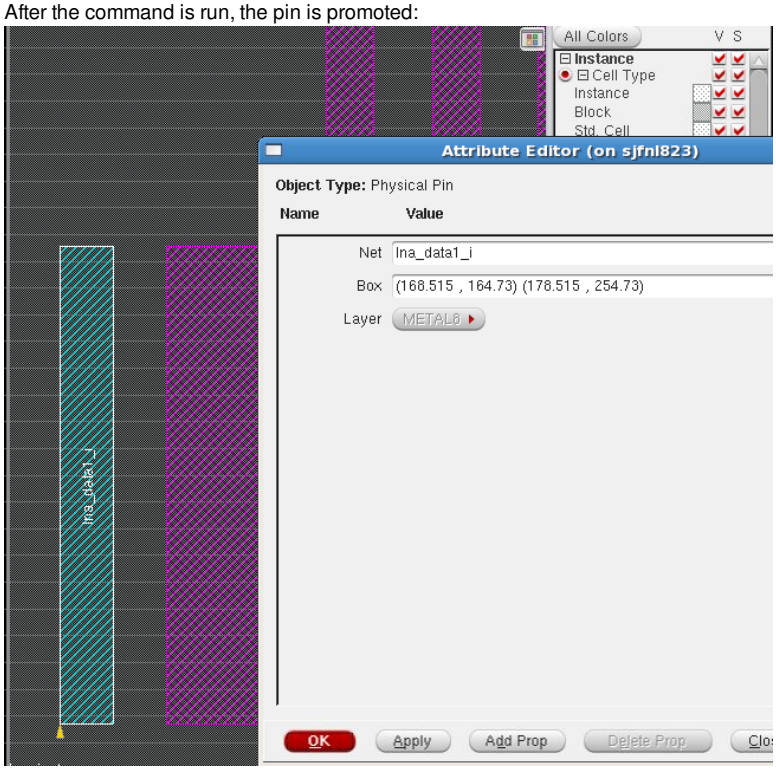
  The following command creates a signal pin for the instance `lna_inst`:

```
create_signal_pin -inst lna_inst
```

  - Before this command is run, the pin is local to `lna_inst`:



- After the command is run, the pin is promoted:



- The following command promotes the specified pins to the top level:

```
create_signal_pin -pins {lna_inst/data1_i lna_inst/data2_i phy/QS_PADP}
```

delete_area_io_fillers

```
delete_area_io_fillers
[-help]
-cell fillerCellName
[-inst fillerInstanceName]
[-prefix prefix]
{-area_io_row_cluster aioRowClusterName | -all_area_io_row_cluster}
```

Deletes area I/O filler cell instances. Use this command after using the `add_area_io_fillers` command.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `delete_area_io_fillers` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man delete_area_io_fillers`
`-all_area_io_row_cluster`	Deletes all area I/O filler cells from all area I/O row clusters. *Data_type*: bool, required
`-area_io_row_cluster aioRowClusterName`	Specifies the area I/O row cluster name where all area I/O filler cells will be deleted. **Note:** This parameter overrides `-all_area_io_row_cluster`, i.e If you specify both, `-area_io_row_cluster aioRowClusterName` and `-all_area_io_row_cluster` parameters, then only `-area_io_row_cluster aioRowClusterName` is valid. *Data_type*: string, required
`-cell fillerCellName`	Specifies the name of the area I/O filler cell to delete. *Data_type*: string, required
`-inst fillerInstanceName`	Specifies the name of the area I/O filler instance name to delete. *Data_type*: string, optional
`-prefix prefix`	Specifies the prefix name of the area I/O filler cell instance to be removed. *Data_type*: string, optional

Example

- The following command deletes a filler cell from all area I/O row clusters:
  ```
  delete_area_io_fillers -cell AO_123 -all_area_io_row_cluster
  ```

Related Information

- `add_area_io_fillers`

delete_bump_connect_target_constraints

```
delete_bump_connect_target_constraints
[-help]
{-bumps list_of_bumps | -selected | -all }
[-io_inst instance_name [-pin_name pin_name [-port_num value]]]
```

Deletes existing properties on specified, selected, or all bumps. If a bump has multiple values for the property, use `-io_inst`, `-pin_name` and `-port_num` to filter out the matched property value for deletion. If only `-io_inst` is specified, then all property values for the pad are deleted. If only `-io_inst` and `-pin_name` are specified, then all property values for the pin are deleted. If the specified property value cannot be found for a bump, Innovus prints an error message for the bump only and continues to process other bumps.

Parameters

`-help`	Prints a brief description that includes type and default information for each `delete_bump_connect_target_constraints` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man delete_bump_connect_target_constraints`
`-all`	Deletes bump connect target constraint on all bumps. *Data_type*: bool, required
`-bumps list_of_bumps`	Deletes bump connect target constraint on specified bumps. The `-bumps` option can accept wild cards. *Data_type*: string list, required
`-io_inst instance_name`	Deletes the bump connect target constraint related to the specified instance. *Data_type*: string, optional
`-pin_name pin_name`	Deletes the bump connect target constraint related to the specified pin. **Note:** This option can be used only if you specify `-io_inst`. *Data_type*: string, optional
`-port_num value`	Deletes the bump connect target constraint related to the specified port number. **Note:** This option can be used only if you specify `-io_inst` and `-pin_name`. *Data_type*: int, optional
`-selected`	Deletes bump connect target constraint on selected bumps. Multiple bumps may be selected. *Data_type*: bool, required

Examples

- The following command removes `bump_connect_target` property on `Bump_1` which pairs with `VDD` PIN of instance `IOPADS_INST/esd`.
  ```
  delete_bump_connect_target_constraints -bumps Bump_1 -io_inst IOPADS_INST/esd -pin_name VDD
  ```
- The following command removes all `bump_connect_target` properties on `Bump_1`.
  ```
  delete_bump_connect_target_constraints -bumps Bump_1
  ```

Related Information

- `create_bump_connect_target_constraint`
- `get_pin_port_number`
- `update_bump_connect_target_constraints`
- Flip Chip Methodologies chapter of the *User Guide*
  - *Port Numbering*
- Tools Menu chapter in the *Menu Reference*

- *Flip Chip*

delete_bumps

```
delete_bumps
[-help]
[-all]
[-bumps bump+]
[-floating]
[-overlap_area_io]
[-overlap_blockages]
[-overlap_keep_out bumpCellName]
[-overlap_macro]
[-selected]
```

Removes bumps from the design. Use this command anytime after creating a bump.

Deletion of assigned bumps is also supported. If any of the bumps being deleted is assigned, `delete_bumps` displays a WARNING message and deletes all specified non-routed bumps, assigned as well as unassigned.

If you click *Undo* (or run the `gui_undo` command) after running `delete_bumps`, the changes made by `delete_bumps` are cancelled out and the following bump properties are recovered:

- Name
- Location
- Port number properties
- Fixed status
- Placement status

On clicking *Redo* (or running the `gui_redo` command), the bump floorplan will revert to the status before undo.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `delete_bumps` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man delete_bumps`
`-all`	Deletes all bumps. *Data_type*: bool, optional
`-bumps bump+`	Specifies the names of the bumps to be deleted. You can use wildcards in the bump name list. *Data_type*: (bump)+, optional
`-floating`	Deletes bumps which are not assigned to any nets. *Data_type*: bool, optional
`-overlap_area_io`	Deletes bumps that overlap with the area I/O cells. You must first select the area I/O cells in the design display area, then use this c *Data_type*: bool, optional
`-overlap_blockages`	Deletes all bumps that overlap routing blockages on the same layer. For example, if a bump on layer M7 overlaps a routing blockag when you use this parameter. *Data_type*: bool, optional
`-overlap_keep_out bumpCellName`	
	Deletes specified bumps that overlap the bump keep out zone. *Data_type*: string, optional
`-overlap_macro`	Deletes all bumps that overlap the selected macros. You must first select the macros in the design display area, then use this comm *Data_type*: bool, optional
`-selected`	Deletes selected bumps. You must first select the bumps in the design display area, then use this command to delete the bumps. *Data_type*: bool, optional

Examples

- The following command deletes all non-routed bumps, assigned as well as unassigned, in the current design:
  ```
  delete_bumps -all
  ```
- The following command deletes all selected bumps, whether assigned or unassigned:
  ```
  delete_bumps -selected
  ```

Related Information

- `assign_bumps`
- `copy_bumps`
- `create_bump`
- `gui_undo`
- `gui_redo`
- `set_bump_master`
- `unassign_bumps`
- `update_bump_name`
- Flip Chip Methodologies chapter in the *User Guide*
- Tools Menu chapter in the *Menu Reference*
  - *Flip Chip*

flip_chip Category Attributes

The root attributes in the `flip_chip` category are:

- flip_chip_allow_layer_change
- flip_chip_allow_routed_bump_edit
- flip_chip_auto_pairing_file
- flip_chip_bottom_layer
- flip_chip_bump_use_octagon_shape
- flip_chip_check_bump_access_directions
- flip_chip_compaction
- flip_chip_connect_power_cell_to_bump
- flip_chip_constraint_file
- flip_chip_drop_via_on_all_shapes
- flip_chip_drop_via_on_power_mesh
- flip_chip_extra_config
- flip_chip_finger_direction
- flip_chip_finger_max_width
- flip_chip_finger_min_width
- flip_chip_finger_target_mesh_layer_range
- flip_chip_honor_bump_connect_target_constraint
- flip_chip_ignore_pad_type_check
- flip_chip_lower_layer_route_width
- flip_chip_lower_layer_prevent_diagonal_routing
- flip_chip_multi_pad_routing_style
- flip_chip_multiple_connection
- flip_chip_pg_mesh_direction
- flip_chip_pg_mesh_main_width
- flip_chip_pg_mesh_max_width
- flip_chip_pg_mesh_width
- flip_chip_prevent_diagonal_wire_access_bumps
- flip_chip_prevent_via_under_bump
- flip_chip_prevent_via_under_bump_extension
- flip_chip_route_pg_style
- flip_chip_route_style
- flip_chip_route_width
- flip_chip_top_layer
- flip_chip_via_abut_bump
- flip_chip_via_to_pad_honor_min_spacing
- flip_chip_wire_to_pad_honor_min_spacing

To set the value of any root attribute, use the `set_db` command:

```
set_db attribute_name value
```

To get the current value of any root attribute, use the `get_db` command:

```
get_db attribute_name
```

To get the current value of all attributes in the `flip_chip` category, use the following command:

```
get_db -category flip_chip
```

flip_chip_allow_layer_change

`flip_chip_allow_layer_change {true | false}`

Default: `false`

Data_type: bool, read/write

Turns on the layer change feature, when set to `true`. Only the PIO mode supports this feature. Both Manhattan and 45-degree routing support this constraint.

Different setting for routing layers will have the different behaviors as shown in the following examples. Assume the top RDL is `TOP_RDL` and the second top RDL is `2nd_RDL`:

- Use `TOP_RDL` as much as possible and `2nd_RDL` is used only when a single layer cannot finish routing in case of cross-over:
  ```
  flip_chip_bottom_layer TOP_RDL flip_chip_top_layer TOP_RDL
  ```
- Freely change layers so that the tool will use the layer resources by its intelligence
  ```
  flip_chip_bottom_layer 2nd_RDL flip_chip_top_layer TOP_RDL
  ```

Related Commands

- `route_flip_chip`

flip_chip_allow_routed_bump_edit

`flip_chip_allow_routed_bump_edit {true | false}`

Default: `true`

Data_type: bool, read/write

Specifies whether Innovus should check routing status of bumps before allowing manipulation. By default, the tool ignores routing status and allows manipulation of routed bumps. Manipulation actions include moving, deleting, re-assigning, and swapping. Set this attribute to `false` if you want the tool to check routing status of bumps and disallow manipulation of routed bumps.

Related Commands

- `assign_bumps`
- `assign_signal_to_bump`
- `copy_bumps`
- `delete_bumps`
- `swap_bump_assignment`
- `unassign_bumps`
- `align_selected`
- `flip_or_rotate_object`
- `move_obj`
- `space_selected`

flip_chip_auto_pairing_file

`flip_chip_auto_pairing_file` *`filename`*

Default: `""`

Data_type: string, read/write

Specifies the output file name to dump out `route_flip_chip` auto pairing information for bump to I/O connection. Depending on the design, the format of the output file can be as follows:

- If the connection between the bump and the instance is known, the format is:

```
create_bump_connect_target_constraint -bumps bump_name -io_inst inst_name
```

- If the connection between the bump and the instance pin is known, the format is:
  ```
  create_bump_connect_target_constraint -bumps bump_name -io_inst inst_name -pin_name pin_name
  ```
- If the connection between the bump and the pin port of the instance is known, the format is:
  ```
  create_bump_connect_target_constraint -bumps bump_name -io_inst inst_name -pin_name pin_name -port_num port_number
  ```

Related Commands

- route_flip_chip
- create_bump_connect_target_constraint

flip_chip_bottom_layer

```
flip_chip_bottom_layer layer_name
```

Default: ""

Data_type: string, read/write

Specifies the bottom-most metal layer by layer number or layer name that the software can use when routing bumps.

Related Commands

- place_peripheral_ios
- route_flip_chip

flip_chip_bump_use_octagon_shape

```
flip_chip_bump_use_octagon_shape {true | false}
```

Default: `false`

Data_type: bool, read/write

Controls how `route_flip_chip` handles special bumps with multiple shapes:

- When this attribute is set to `true`, the tool uses an octagonal shape to mimic the special bump.
- When this attribute is set to `false`, the tool:
  - Uses the bump's exact LEF description if the bump is simple enough.
  - May replace the bump shape with an octagonal shape if the bump is not simple according to the tool's internal criterion.

Note: When the special bump is replaced with an octagonal shape, the routing quality may be improved, but there may be DRC violations.

Related Commands

- route_flip_chip

flip_chip_check_bump_access_directions

```
flip_chip_check_bump_access_directions {true | false}
```

Default: `false`

Data_type: bool, read/write

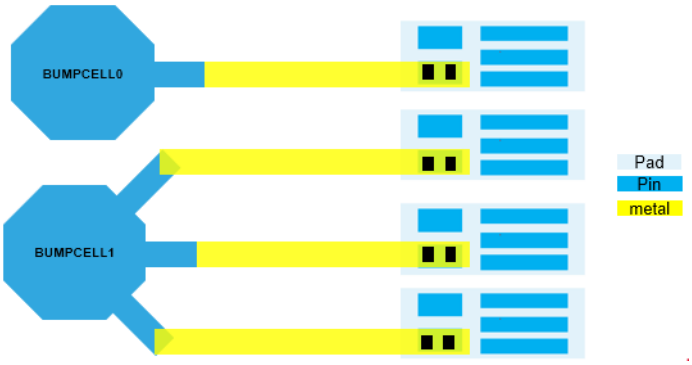
Specifies whether `route_flip_chip` should check the access direction of bump cells. Use this attribute if you have an octagonal bump pad structure with a pre-drawing wire for outside access, where the pre-drawing wire is a pin in the bump LEF.

- When this attribute is set to `true`, `route_flip_chip` checks the bump LEF to detect the bump access direction and then connects the pad to the bump pre-routed wire based on the detected direction. If the pre-routed wire is a 45-degree wire, the `manhattan` routing style is not supported.
  Note that one bump cell can have multiple access directions.
- When this attribute is set to `false`, `route_flip_chip` skips checking bump access direction.

Note: This attribute is applicable only for `route_flip_chip -target connect_bump_to_pad`. If this option is used for `route_flip_chip -target connect_bump_to_ring_stripe`, `route_flip_chip` will ignore the setting and display a warning.

In the example below, `route_flip_chip` checks the bump LEF to detect the bump access directions and then connects the pads to the bump pre-routed wires based on the detected directions:

```
> set_db flip_chip_check_bump_access_directions true
> route_flip_chip -target connect_bump_to_pad -route_engine global_detail
```



Related Commands

- `route_flip_chip`

flip_chip_compaction

`flip_chip_compaction {true | false}`

Default: `false`

Data_type: bool, read/write

Turns on compaction routing, when set to `true`.

In the flip chip post-route stage, the compaction function pushes the routing as close as possible to bumps in order to leave more routing resource for subsequent routing steps, such as P/G bump connections or general power routing.

Note: This feature cannot be used with resistance driven and diffpair routings because the compaction action may change the wire length and routing pattern. This feature does not support two-layer RDL routing.

Related Commands

- `place_peripheral_ios`
- `route_flip_chip`

flip_chip_connect_power_cell_to_bump

`flip_chip_connect_power_cell_to_bump {true | false}`

Default: `false`

Data_type: bool, read/write

Connects all power bumps to the I/O cell pin. It does not connect to a power or ground stripe or ring.

When used with the `route_flip_chip -target signal` parameter, you can connect power bumps to I/O power pads. To make this type of connection, you need the following information:

- The Verilog netlist must contain the I/O power pad, for example:
  ```
  VDDCELL VDD_INST (.PAD());
  ```
- The LEF macro pin must contain the USE POWER statement, for example:
  ```
  MACRO VDDCELL
  PIN PAD
  ```

```
    USE POWER;
```

To improve the quality of routing in situations where there may be many power bumps in the design, you can pair power bumps to power cells by using this attribute in conjunction with the `flip_chip_constraint_file` attribute. For example:

```
set_db flip_chip_connect_power_cell_to_bump true
```

```
set_db flip_chip_constraint_file pair.const
```

where `pair.const` specifies the following information:

```
### Syntax: net pad [pad2] bump [bump2] ###
```

```
PAIR
```

```
VDD IOPADS_INST/esd Bump_90_9_8
```

```
END PAIR
```

Related Commands

- `place_peripheral_ios`
- `route_flip_chip`

flip_chip_constraint_file

```
flip_chip_constraint_file file_name
```

Default: `""`

Data_type: string, read/write

Specifies the file that contains constraints for flip chip routing. For more information, see "*Useful Constraints for Flip Chip Routing*" in the Flip Chip Methodologies chapter of the *User Guide.*

To set the constraint file, you can use this attribute as follows:

```
set_db flip_chip_constraint_file pair.const
```

where `pair.const` specifies the following information:

```
### Syntax: net pad [pad2] bump [bump2] ###
```

```
PAIR
```

```
VDD IOPADS_INST/esd Bump_90_9_8
```

```
END PAIR
```

Related Commands

- `place_peripheral_ios`
- `route_flip_chip`

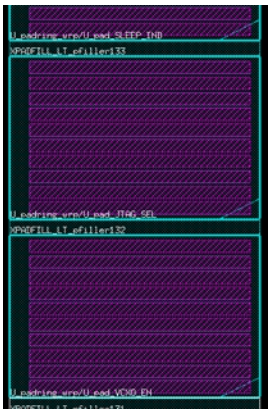
flip_chip_drop_via_on_all_shapes

```
flip_chip_drop_via_on_all_shapes {true | false}
```

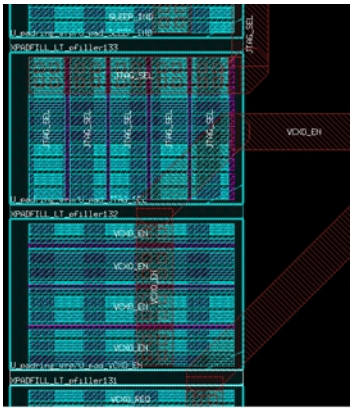
Default: `false`

Data_type: bool, read/write

Drops vias on all defined geometries. Consider the pin shape below:



For the given shape, the routing result will be as follows when `flip_chip_drop_via_on_all_shapes` is set to `true`:



Related Commands

- `place_peripheral_ios`
- `route_flip_chip`

flip_chip_drop_via_on_power_mesh

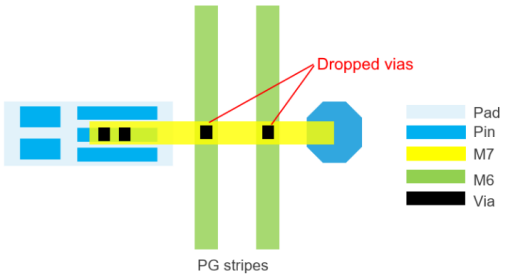
`flip_chip_drop_via_on_power_mesh` *`layer_name`*

Default: `""`

Data_type: string, read/write

Specifies the number or name of the metal layer that can be used by the tool for dropping a via on a PG mesh. `route_flip_chip` automatically drops a via on the PG mesh on the specified metal layer if the PG RDL route crosses the PG mesh. For example:

```
set_db flip_chip_drop_via_on_power_mesh M6
set_db flip_chip_connect_power_cell_to_bump true
route_flip_chip -target connect_bump_to_pad -route_engine global_detail
```



The via is dropped only if there is no DRC violation due to the via drop.

The `flip_chip_drop_via_on_power_mesh` attribute does not affect the routing results from PG bumps to pad.

Related Commands

- `place_peripheral_ios`
- `route_flip_chip`

flip_chip_extra_config

`flip_chip_extra_config` *`file_name`*

Default: ""

Data_type: string, read/write

Specifies the name of an extra configuration file.

Related Commands

- `place_peripheral_ios`
- `route_flip_chip`

flip_chip_finger_direction

`flip_chip_finger_direction {n|e|s|w}`

Default: `n`

Data_type: enum, read/write

Specifies the main expansion direction for PG finger routing. The default direction is North.

Related Commands

- `place_peripheral_ios`
- `route_flip_chip`

flip_chip_finger_max_width

`flip_chip_finger_max_width` *`value`*

Default: `0.0`

Data_type: double, read/write

Specifies the maximum routing width for one wire segment of the PG finger. By default, there is no constraint on the max width; `route_flip_chip` uses `MAXWIDTH` in LEF. If this constraint is not specified in LEF, `route_flip_chip` uses the maximum width possible.

If there is a conflict between the user-specified constraint and the LEF rule, the LEF rule is used as the maximum width.

Related Commands

- `place_peripheral_ios`
- `route_flip_chip`

flip_chip_finger_min_width

`flip_chip_finger_min_width` *`value`*

Default: `0.0`

Data_type: double, read/write

Specifies the minimum routing width for one wire segment of the PG finger. The default is the RDL minimum width defined in LEF. If there is a conflict between the user-specified constraint and the LEF rule, the LEF rule is used as the minimum width.

Related Commands

- `place_peripheral_ios`
- `route_flip_chip`

flip_chip_finger_target_mesh_layer_range

`flip_chip_finger_target_mesh_layer_range` *`layer_range`*

Default: `""`

Data_type: string, read/write

Specifies the target mesh layer range of the power mesh that will be connected for finger routing. If not specified, no layer of the power mesh will be connected.

Related Commands

- `place_peripheral_ios`
- `route_flip_chip`

flip_chip_honor_bump_connect_target_constraint

`flip_chip_honor_bump_connect_target_constraint {true | false}`

Default: `false`

Data_type: bool, read/write

Enables port numbering routing, when set to `true`.

Related Commands

- `place_peripheral_ios`
- `route_flip_chip`

flip_chip_ignore_pad_type_check

`flip_chip_ignore_pad_type_check {true | false}`

Default: `true`

Data_type: bool, read/write

Specifies whether or not CLASS PAD cells are supported in the flip chip flow.

Related Commands

- `place_peripheral_ios`
- `route_flip_chip`

flip_chip_lower_layer_route_width

`flip_chip_lower_layer_route_width` *`value`*

Default = `0`, *Min* = `0.000000`, *Max* = `10000000000.000000`

Data_type: double, read/write

Specifies a different value for lower routing layer. Unit is in microns.

Related Commands

- `place_peripheral_ios`
- `route_flip_chip`

flip_chip_lower_layer_prevent_diagonal_routing

`flip_chip_lower_layer_prevent_diagonal_routing {true | false}`

Default: `false`

Data_type: bool, read/write

Prevents diagonal routing for lower layer, when set to `true`.

Related Commands

- `place_peripheral_ios`
- `route_flip_chip`

flip_chip_multi_pad_routing_style

`flip_chip_multi_pad_routing_style {default serial star}`

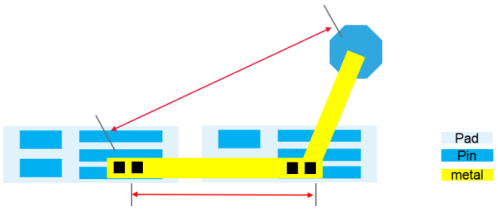
Default: `default`

Data_type: enum, read/write

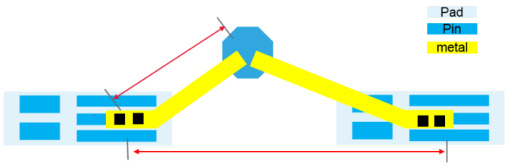
Specifies the routing style for multi-pad routing:

- `default`: The flip chip router (`route_flip_chip`) connects a bump to multiple pads with multiple different wires as shown below:

- `serial`: `route_flip_chip` connects a bump first to one pad and then to the other pad, as shown below, provided that the distance from the bump to either pad is more than the distance between the two pads:

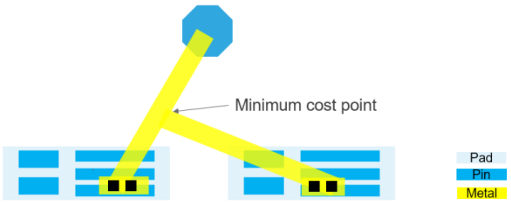


If the distance from the bump to any pad is shorter than the distance between the two pads, `route_flip_chip` reverts to the default behavior of multiple wire connection even if `flip_chip_multi_pad_routing_style` is set to `serial`.



This mode supports only two-pad routing.

- `star`: `route_flip_chip` connects a bump first to the closest pad. It then finds a minimum cost point on its routing path and connects this point to another pad as shown below:



The minimum cost point could be anywhere on the first routing path, including its beginning and end.

Both the `serial` and `star` modes are useful for saving routing resources in complex designs as they share the routing between pads. However, the `serial` mode supports only two-pad routing. Use the `star` mode if you want to route more than two pads.

Note: The `flip_chip_multi_pad_routing_style` works only if the `flip_chip_multiple_connection multiple_pads_to_bump` is specified. For example, to use the `star` mode, you should specify the following commands before running `route_flip_chip`:

```
set_db flip_chip_multiple_connection multiple_pads_to_bump
set_db multi_pad_routing_style star
```

Related Commands

- `place_peripheral_ios`
- `route_flip_chip`

flip_chip_multiple_connection

`flip_chip_multiple_connection {multiple_pads_to_bump | multiple_bumps_to_pad | default}`

Default: `default`

Data_type: enum, read/write

Specifies routing connections between multiple pads and bumps. You can specify one of the following options:

- `multiple_pads_to_bump`
  Enables routing from multiple pads to one bump in parallel. This option is supported by both AIO and PIO routing styles.
- `multiple_bumps_to_pad`
  Enables routing from multiple bumps to one pad in parallel. This option is supported by both AIO and PIO routing styles.
- `default`
  Enables routing from one bump to one pad in parallel. This means there is no multiple connection.

Related Commands

- `place_peripheral_ios`
- `route_flip_chip`

flip_chip_pg_mesh_direction

`flip_chip_pg_mesh_direction {default | horizontal | vertical}`

Default: `default`

Data_type: enum, read/write

Specifies the routing direction of the PG mesh. You can specify one of the following values for this option:

- `default` - Uses the preferred routing direction defined in the Tech LEF.
- `horizontal` - Uses the horizontal routing direction.
- `vertical` - Uses the vertical routing direction.

The `flip_chip_pg_mesh_direction` attribute can be used only if `flip_chip_route_pg_style` is set to `mesh`.

Related Commands

- route_flip_chip

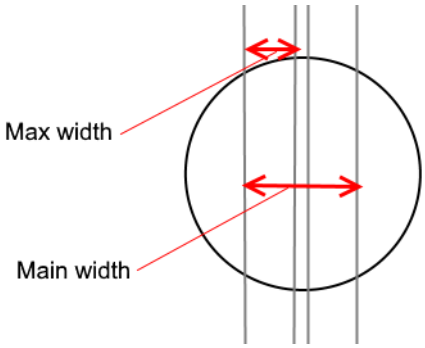
flip_chip_pg_mesh_main_width

`flip_chip_pg_mesh_main_width value`

Default: `0.0`; *Min*: `0`; *Max*: `10000000000`

Data_type: double, read/write

Specifies the main width of the PG mesh. This attribute can be used only if `flip_chip_route_pg_style` is set to `mesh`.



Related Commands

- route_flip_chip

flip_chip_pg_mesh_max_width

`flip_chip_pg_mesh_max_width value`

Default: `0.0`; *Min*: `0`; *Max*: `10000000000`

Data_type: double, read/write

Specifies the maximum width of the routing wire in a PG mesh. This attribute can be used only if `flip_chip_route_pg_style` is set to `mesh`.

By default there is no constraint on the maximum width. `fcroute` uses `MAXWIDTH` in the Tech LEF. If there is a conflict between the value you specify and the LEF rule, the tool uses the LEF rule as the maximum width.

Related Commands

- route_flip_chip

flip_chip_pg_mesh_width

`flip_chip_pg_mesh_width value`

Default: `""`

Data_type: string, read/write

Specifies the exact routing width of the PG mesh. If only one value is specified with `-pg_mesh_width`, the value is applicable for the width of both horizontal and vertical wires. If two values are specified, the first value is applicable for the width of horizontal wires and the second value is applicable for the width of vertical wires.

This attribute can be used only if `flip_chip_route_pg_style` is set to `mesh`.

Related Commands

- route_flip_chip

flip_chip_prevent_diagonal_wire_access_bumps

`flip_chip_prevent_diagonal_wire_access_bumps {true | false}`

Default: `false`

Data_type: bool, read/write

Specifies whether or not diagonal wires can be used to access bumps.

Related Commands

- `route_flip_chip`

flip_chip_prevent_via_under_bump

`flip_chip_prevent_via_under_bump {true | false}`

Default: `false`

Data_type: bool, read/write

Specifies whether or not vias are generated directly under the bump. If you set this parameter to `true`, the vias generated are prevented from connecting directly to the bump.

This option is available for both signal and power bumps.

Related Commands

- `place_peripheral_ios`
- `route_flip_chip`

flip_chip_prevent_via_under_bump_extension

`flip_chip_prevent_via_under_bump_extension` *`value`*

Default: `0.0`

Data_type: double, read/write

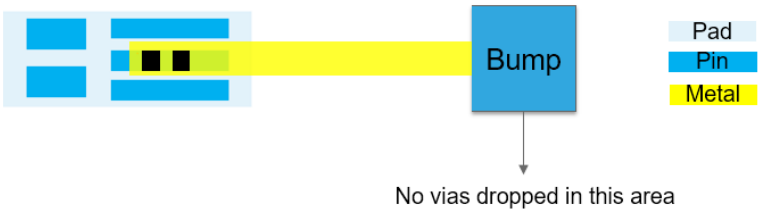
Specifies a spacing value to define an extension area around the bump in which vias should not be dropped. The spacing value must be greater than or equal to `0`.

This option can be used only if `flip_chip_prevent_via_under_bump` is set to `true`.

The following examples illustrate how the `flip_chip_prevent_via_under_bump` and `flip_chip_prevent_via_under_bump_extension` attributes can be used to control the spacing around bumps when vias are dropped:

- In the example below, no vias are dropped directly under the bump as `flip_chip_prevent_via_under_bump` is set to `true`:

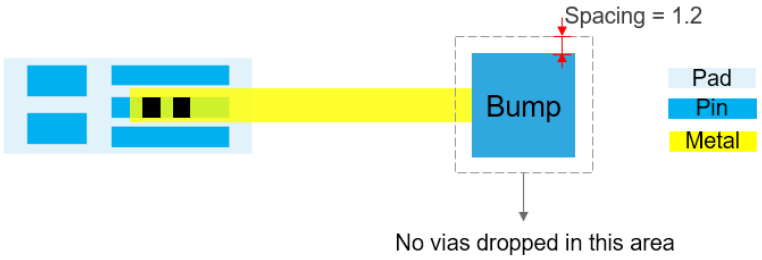
```
> set_db flip_chip_prevent_via_under_bump true
> route_flip_chip -target connect_bump_to_pad -route_engine global_detail
```



  Additionally, if `flip_chip_prevent_via_under_bump_extension` is set to `0`, the same result is achieved.

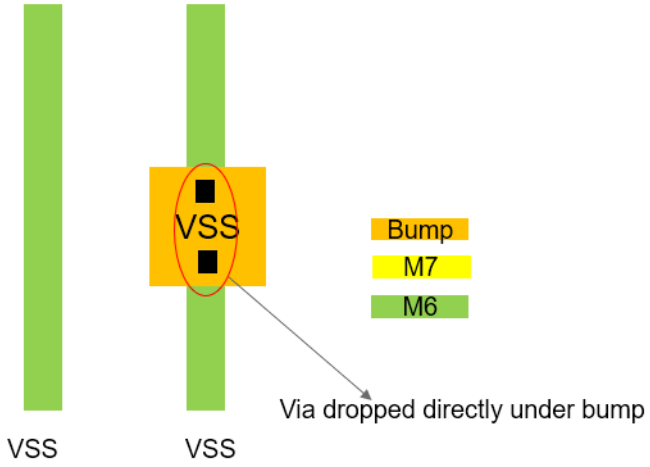
- In the example below, no vias are dropped in the extension area defined around the bump:

```
> set_db flip_chip_prevent_via_under_bump true
> set_db flip_chip_prevent_via_under_bump_extension 1.2
> route_flip_chip -target connect_bump_to_pad -route_engine global_detail
```



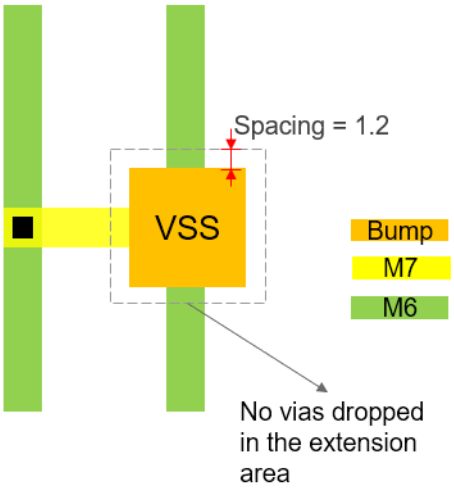
- In the example below, vias are dropped directly under the power bumps as `flip_chip_prevent_via_under_bump` is set to `false`:

```
> set_db flip_chip_prevent_via_under_bump false
> route_flip_chip -target connect_bump_to_ring_stripe
```



- In the example below, no vias are dropped in the extension area defined around the power bumps:

```
> set_db flip_chip_prevent_via_under_bump true
> set_db flip_chip_prevent_via_under_bump_extension 1.2
> route_flip_chip -target connect_bump_to_ring_stripe
```



Related Commands

- `route_flip_chip`

flip_chip_route_pg_style

`flip_chip_route_pg_style {none finger mesh}`

Default: `none`

Data_type: enum, read/write

Specifies the routing pattern for flip chip route power routing:

- `none`: For normal PG routing
- `finger`: For PG finger routing style, in which you need to specify the routing direction. This option must be used with `route_flip_chip -region` option.
- `mesh`: For mesh-style PG routing on the redistribution layer (RDL). This option is used to increase PG density. If you specify PG mesh routing, you must specify the main width of the PG mesh by using the `flip_chip_pg_mesh_main_width` parameter.

Related Commands

- `place_peripheral_ios`
- `route_flip_chip`

flip_chip_route_style

`flip_chip_route_style {manhattan | diagonal}`

Default: `diagonal`

Data_type: enum, read/write

Determines the type of routing, either `manhattan` or `diagonal`.

Note: `diagonal` routing style does not work in conjunction with the following `route_flip_chip` parameter:

- `-double_bend_route`

Related Commands

- `place_peripheral_ios`
- `route_flip_chip`

flip_chip_route_width

`flip_chip_route_width` *`value`*

Default: `0.0`

Data_type: double, read/write

Connects wires to bumps using the specified route width not wider than the I/O pad pin.

Related Commands

- `place_peripheral_ios`
- `route_flip_chip`

flip_chip_top_layer

`flip_chip_top_layer` *`layer_name`*

Default: ""

Data_type: string, read/write

Specifies the top-most metal layer by layer number or layer name that the software can use when routing bumps.

Related Commands

- `place_peripheral_ios`
- `route_flip_chip`

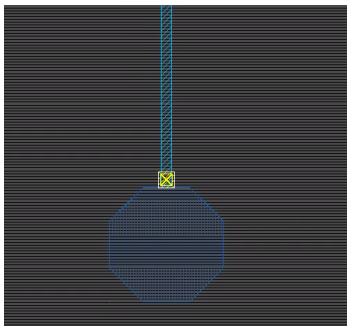
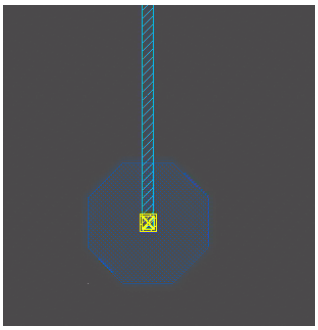
flip_chip_via_abut_bump

`flip_chip_via_abut_bump {true | false}`

Default: `false`

Data_type: bool, read/write

Specifies that via should abut the bump only if routing can be done successfully.

With `flip_chip_via_abut_bump` set to `true`	Without `flip_chip_via_abut_bump` set to `true`
	 In this case, via is dropped inside bump as shown above. This is the default setting.

Related Commands

- `route_flip_chip`

flip_chip_via_to_pad_honor_min_spacing

`flip_chip_via_to_pad_honor_min_spacing` *`value`*

Default: `0.0`

Data_type: double, read/write

Specifies the minimum spacing between the bump/pad and a via.

Related Commands

- `route_flip_chip`

flip_chip_wire_to_pad_honor_min_spacing

`flip_chip_wire_to_pad_honor_min_spacing` *`value`*

Default: `0.0`

Data_type: double, read/write

Specifies the minimum spacing between the bump/pad and a wire.

Related Commands

- `route_flip_chip`

get_bond_pad

```
get_bond_pad
[-help]
-io_inst InstName
[-pin pinName]
```

Gets the current stagger position of a bond pad on a specified I/O instance.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `get_bond_pad` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man get_bond_pad`
`-io_inst instName`	Specifies the name of the I/O instance whose bond pad position is reported. *Data_type*: string, required
`-pin pinName`	Specifies the name of the pin inside the instance. *Data_type*: string, optional

Example

- The following command gets the current stagger position of the bond pad on I/O instance `pad_addr[30]`:
  ```
  get_bond_pad -io_inst {pad_addr[30]}
  ```
  Returns:
  ```
  Bond Pad Name = BUMPIN_337, Position = I
  ```

Related Information

- `check_bond_pad_spacing`
- `place_bond_pad`
- `place_bond_pad_by_stagger`
- `reset_fixed_bond_pad`
- `set_bond_pads_to_fixed`
- `space_bond_pads`

get_physical_info

```
get_physical_info
[-help]
[-highlight]
{-obj string | -selected }
[-shapes {ring stripe followpin iowire corewire blockwire padring blockring fillwire fillwireopc drcfill none}]
[-user_class {subclass_name_list}]
[-type {connection | routing_path}]
```

Gets the physical connection or routing path between IO pads and bumps. For example, you can select a bump or specify bump name to get information about connected IO pad. Alternatively, you can select or specify an IO pad and get information about connected bumps. You can specify IO pads by pad name, instancen:pin combination, or instance:pin:port combination.

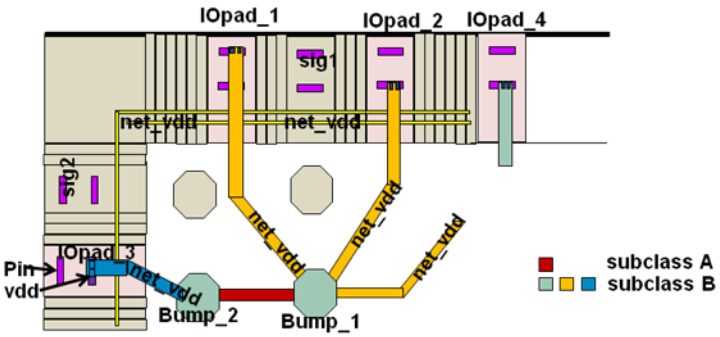
Use this command to check connections after RDL routing, either manual or through `route_flip_chip`.

Parameters

`-help`	Prints a brief description that includes type and default information for each `get_physical_info` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man get_physical_info`
`-highlight`	Highlights the routing path. *Data_type*: bool, optional
`-obj string`	Specifies the object from which to get the physical information. For bump, specify the complete bump name.For IO pad, specify one of the following: • *instance_name* • *instance_name:pin_name* • *instance_name:pin_name:port_number* *Data_type*: string, optional
`-selected`	Gets the physical information for the selected object. *Data_type*: bool, optional
`-shapes {ring stripe followpin iowire corewire blockwire padring blockring fillwire fillwireopc drcfill none}]`	
	Specifies the shapes of the wires and vias to be checked. The default value is `iowire`. *Data_type*: (enum)+, optional
`-type {connection | routing_path}`	
	Specifies the type of the physical information to return. The default type is `connection`. • `connection`: Returns the objects that are physically connected to the object you have specified or selected. For example, if you have selected an IO pad, it returns the names of the bump that are connected with the IO pad. If you have selected or specified a bump, it returns the IO pads connected to the bump, using the format *`ioInst:pin:port`*. • `routing_path`: Returns the pointer list of the wires and vias that are physically connected to the object you have selected or specified. *Data_type*: enum, optional
`-user_class {subclass_name_list}`	Specifies the subclass name of the wires and vias to be checked. You can specify multiple subclass names enclosed in braces. You can also use wildcards while specifying subclass names. *Data_type*: string, optional

Examples

Consider the following design.



For this design, `get_physical_info` reports information as follows:

- `get_physical_info -obj {IOpad_1:vdd:1}`
  Returns `{Bump_1}`
- `get_physical_info  -type connection -obj {Bump_1}`
  Returns `{IOpad_1:vdd:1 IOpad_2:vdd:2  Bump_2}`
- `get_physical_info -obj {Bump_1} -user_class B`
  Returns `{IOpad_1:vdd:1 IOpad_2:vdd:2}`
- `get_physical_info -type routing_path -obj {Bump_2}`
  Returns `{0x2aaaba6c0ab0 0x2aaaba6c0b40 0x2aaaba6c0b70}`

Related Information

- `route_flip_chip`
- Flip Chip Methodologies chapter in the *User Guide*

get_pin_port_number

```
get_pin_port_number
[-help]
[-bumps bump_list]
[-geometry_height {min [max]}]
[-geometry_width {min [max]}]
[-layer string]
[-region x1 y1 x2 y2]
{[-io_inst_cell cell_name | -io_inst instance_name] [-selected]}
{-pin pin_name | -net net_name}
```

Determines suitable ports available for a bump. The command has the following functions:

- Check if the IO pad has a standard LEF definition.
- Suggest suitable ports for flip chip routing on the specified instance based on the pin name or nets in a specified area.
- Filter ports by layer, geometry size, and area. Only the ports that match the specified filter rules are returned by `get_pin_port_number`.
- Consider routing between bumps and ports to return the combinations of `{bump:inst:pin:port}` if the bumps for which ports are to be found are specified.

`get_pin_port_number` returns a port number string list. You can use `set result [get_pin_port_number]` or `catch {get_pin_port_number} result` to get the result.

The following commands provide additional support for the port numbering feature. Use these commands to add, edit, and delete bump properties easily:

- create_bump_connect_target_constraint
- delete_bump_connect_target_constraints
- update_bump_connect_target_constraints

Parameters

`-help`	Prints a brief description that includes type and default information for each `get_pin_port_number` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man get_pin_port_number`
`-bumps` *bump_list*	Finds suitable ports for the specified bumps based on the smallest distance. This option supports only 1:1 pairing between ports and bumps. Therefore, if the number of ports is different from the number of bumps, pairing is based on the smaller number. `get_pin_port_number` allows empty bumps or bumps assigned to the same net as the pin. If the assigned net is different from the net with which the pin is connected, `get_pin_port_number` issues an error message and quits. *Data_type*: string, optional
`-geometry_height {`*min* `[`*max*`]}`	
	Specifies the minimum and maximum values for the height of the target port's geometry. If you specify only one value, the height is taken as equal to the specified value. Only ports that match the specified height criterion are returned by `get_pin_port_number`. *Data_type*: string, optional
`-geometry_width {`*min* `[`*max*`]}`	
	Specifies the minimum and maximum values of the width of the target port's geometry. If you specify only one value, the width is taken as equal to the specified value. Only ports that match the specified width criterion are returned by `get_pin_port_number`. *Data_type*: string, optional
`-io_inst` *instance_name*	
	Finds pin port number of the specified instance. Wild cards are supported in *instance_name*. *Data_type*: string, required
`-io_inst_cell` *cell_name*	

	Finds the pin port number of the instance of the specified cell. Wild cards are supported in *cell_name*. *Data_type*: string, required
-layer *string*	Filters target ports by layer. If you specify this parameter, only ports on the specified layer are returned. You can specify either layer name or layer ID, such as M8. *Data_type*: string, optional
-net *net_name*	Finds the port number on the pin connected to the specified net. *Data_type*: string, required
-pin *pin_name*	Finds the port number on the specified pin. *Data_type*: string, required
-region *x1 y1 x2 y2*	Specifies the area in which the command will be used. findPInPortNumber returns port numbers overlapping with the specified area. *Data_type*: rect, optional

-selected	Finds suitable ports for the selected instance and bumps:
	• If only bumps are selected, get_pin_port_number first finds the ports satisfying the specified filtering rules and then considers the routing between these ports and the selected bumps.
	• If only IO instances are selected:
	  ◦ If only one IO instance is selected, get_pin_port_number finds the ports satisfying the specified filtering rules.
	  ◦ If more than one IO instance is selected, get_pin_port_number issues an error message stating that only a single IO instance is supported for this option and then quit.
	• If both IO instances and bumps are selected:
	  ◦ If only one IO instance is selected, get_pin_port_number first finds the ports satisfying the specified filtering rules and then considers the routing between these ports and the selected bumps.
	  ◦ If more than one IO instance is selected, get_pin_port_number issues an error message stating that only a single IO instance is supported for this option and then quit.
	Data_type: bool, required

Examples

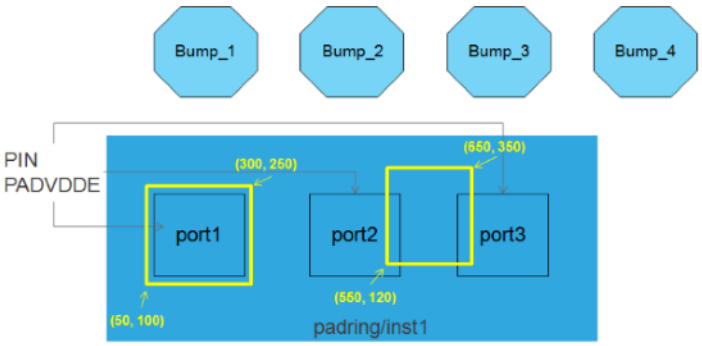
- The following command suggests port numbers on instances named `IOPADS_INST/esd*` for the pins connected to net `VDD`.

  ```
  get_pin_port_number -io_inst IOPADS_INST/esd* -net VDD
  ```

  The tool returns a list of suitable port numbers as follows:

  ```
  IOPADS_INST/esd:VDD:1 IOPADS_INST/esd1:VDD:1
  ```

- The diagram below depicts an IO instance `padring/inst1` with three ports for `PIN PADVDDE` on the same layer and with the same size. The bumps `Bump_1`, `Bump_2`, `Bump_3`, and `Bump_4` are empty.



  - The following command finds ports of the pin `PADVDDE` on the instance `padring/inst1` that meet the specified area constraint:

    ```
    get_pin_port_number -io_inst padring/inst1 -pin PADVDDE -region "50 100 300 250"
    ```

    For the instance depicted in the diagram above, the command returns `{padring/inst1:PADVDDE:1}`.

  - The following command finds ports of the pin `PADVDDE` on the instance `padring/inst1` that meet the specified area constraint:

    ```
    get_pin_port_number -instName padring/inst1 -pinName PADVDDE -area "550 120 650 350"
    ```

    For the instance depicted, the command returns `{padring/inst1:PADVDDE:2 padring/inst1:PADVDDE:3}`. Both `port2` and `port3` are returned as they overlap with the specified area constraint.

  - The following command finds suitable ports of the pin `PADVDDE` on the instance `padring/inst1` for the specified bump:

    ```
    get_pin_port_number -instName padring/inst1 -pinName PADVDDE -bumps Bump_2
    ```

    For the instance depicted, the command returns `{Bump_2:padring/inst1:PADVDDE:2}`.

  - The following command finds suitable ports of the pin PADVDDE on the instance padring/inst1 for the specified bumps:

    ```
    get_pin_port_number -instName padring/inst1 -pinName PADVDDE -bumps {Bump_1 Bump_2 Bump_3 Bump_4}
    ```

    For the instance depicted, the command returns `{Bump_1:padring/inst1:PADVDDE:1 Bump_2:padring/inst1:PADVDDE:2 Bump_3:padring/inst1:PADVDDE:3}`.

Related Information

- `create_bump_connect_target_constraint`
- `delete_bump_connect_target_constraints`
- `update_bump_connect_target_constraints`
- Flip Chip Methodologies chapter of the *User Guide*
  - *Port Numbering*
- Tools Menu chapter in the *Menu Reference*
  - *Flip Chip*

gui_show_bump_connections

```
gui_show_bump_connections
[-help]
[-honor_color]
[-multi_bumps_to_pad]
[-multi_pads_to_bump]
[-remove]
[[[-net net_list] [-bump bump_list] [-io_inst io_inst_list] [-selected]]|[[-bump_type {all | power | signal}] [-target {pad | ring}]]]
```

Displays the connection as a flightline between a power/ground (PG) bump and a PG I/O pad, during signal bump assignment. You can remove the flightlines between the bumps and I/O pads by using `gui_show_bump_connections -remove`.

For both signal and power pins, `gui_show_bump_connections`, by default, displays a long pin connection with the correct location, instead of the center of the pin.

Note: `gui_show_bump_connections` honors the port tag information in bumps for both signal and power flightline for multiple pad and multiple bump routing.

Parameters

`-help`	Prints a brief description that includes type and default information for each `gui_show_bump_connections` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man gui_show_bump_connections`
`-bump_type {all | power | signal}`	
	Specifies the connecting bump type, which could be: • `all` - All the bump types • `power` - Power bump • `signal` - Signal bump *Default*: `all` *Data_type*: enum, optional
`-bumps` *bump_list*	Specifies bumps for which flightlines are to be displayed. *Data_type*: string, optional
`-honor_color`	Colors flightlines according to the settings defined with the `read_bump_color_map_file` command. *Data_type*: bool, optional
`-io_insts {`*io_inst_list*`}`	
	Specifies IO instances or blocks for which flightlines are to be displayed. Specified IO instances should be CLASS PAD AREAIO instances. Specified blocks should have at least one pin port in CLASS BLOCK cell containing the CLASS BUMP property. *Data_type*: string, optional
`-multi_bumps_to_pad`	
	Connects multiple bumps to one pad. *Data_type*: bool, optional
`-multi_pads_to_bump`	
	Connects multiple pads to one bump. *Data_type*: bool, optional
`-nets` *net_list*	Specifies nets for which flightlines are to be displayed. *Data_type*: string, optional
`-remove`	Removes the bump connections. *Data_type*: bool, optional
`-selected`	Displays flightlines for selected bumps and IO instances in bold. *Data_type*: bool, optional
`-target {pad | ring}`	
	Specifies the connecting target, which could be a `pad` or a `ring`. *Data_type*: enum, optional

Examples

The following command displays flightlines for bumps `Bump_1` and `Bump_2`.

```
gui_show_bump_connections -bumps {Bump_1 Bump_2}
```

Related Information

- `read_bump_color_map_file`
- `route_flip_chip`
- Attributes in the `flip_chip` category
- Flip Chip Methodologies chapter of the *User Guide*
  - *Viewing Flip Chip Flightlines*
- Tools Menu chapter of the *Menu Reference*
  - *Flip Chip*

place_bond_pad

```
place_bond_pad
[-help]
[-bond_pad padName]
[-fix]
{-io_inst instName | -selected}
[-pin pinName]
[-position {inner middle outer}]
```

Places a bond pad on a specified I/O instance. It defines the stagger position of a specified bump on a single or selected set of I/O instances.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `place_bond_pad` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man place_bond_pad`
`-bond_pad` *padName*	Specifies the name of the bond pad. *Data_type*: string, optional
`-fix`	Assigns a fixed status to the specified bond pad. This restricts other bond pad commands from modifying the bond pad. *Data_type*: bool, optional
`-io_inst` *name*	Specifies the name of the I/O instance of where the bond pad will be placed. *Data_type*: string, required
`-pin` *pinName*	Specifies the name of the pin inside the instance. *Data_type*: string, optional
`-position {inner middle outer}`	
	Defines the stagger position of the specified bond pad on the specified I/O instance. You can stagger the position of the bond pad using the `inner`, `middle`, and `outer` positions. *Data_type*: enum, optional
`-selected`	Places a selected bond pad on the specified I/O instance. You must first select the bond pad in the design display area, then use this parameter to place the bond pad on the specified I/O instance. *Data_type*: bool, required

Related Information

- `check_bond_pad_spacing`
- `get_bond_pad`
- `place_bond_pad_by_stagger`
- `reset_fixed_bond_pad`
- `set_bond_pads_to_fixed`
- `space_bond_pads`

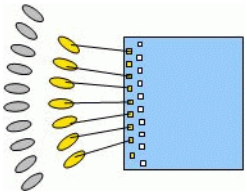
place_bond_pad_by_stagger

```
place_bond_pad_by_stagger
[-help]
[-bond_pad position padName position padName...]
{-pattern pattern_string | -start_stagger_position {inner middle outer}}
[-pin pinName]
{{-first_io_inst_name instName [-last_io_inst instName]} | {[-side {east west north south}] [-ring number]} | -all}
```

Generates a staggered wirebond pad pattern on top of an existing I/O ring. In this design methodology, bond pads are separate LEF macro definitions from the I/O driver cells, but are placed on top of the I/O driver cells to form the combined bond pad and I/O driver circuitry.

The command only supports `CLASS PAD` instances.

To produce the staggered pattern, multiple bond pad cells are defined in the LEF for each
I/O cell, each with its bond pad in a specific location corresponding to an outer, middle, or inner stagger position. The bond pads are defined such that when placed on the I/O cell, any one of them will connect to the I/O cell pin with no or minimal RDL routing.



Innovus generates a stagger pattern by selecting the outer, middle, or inner stagger bond pad to place on each I/O cell based on the location of that I/O instance in the I/O ring and the pattern that is being generated. If the order of the I/O cells is changed, or if cells are added or deleted, the stagger pattern may be regenerated. Because the bond pad macros are separately defined, they can be easily swapped with bond pads of different stagger positions without changing the logic of the design.

To define the relationship between the bond pads and I/O cells in the LEF file,
the following properties must be defined in each LEF file where they are referenced:

```
PROPERTYDEFINITIONS

     PIN bondPadOuter STRING;

     PIN bondPadMiddle STRING;

     PIN bondPadInner STRING;

     PIN ioCellOriginX REAL;

     PIN ioCellOriginY REAL;

     MACRO ioCellOffsetX REAL;

     MACRO ioCellOffsetY REAL;

 END PROPERTYDEFINITIONS
```

The LEF macros define the property values as follows:

```
MACRO IOCELL

     CLASS PAD ;

     ...

     SIZE 50.0 BY 225 ;

     ...

     PIN PAD

         PROPERTY bondPadOuter "BUMPA BUMPA1 BUMPD" ;

         PROPERTY bondPadMiddle "BUMPB" ;

         PROPERTY bondPadInner "BUMPC" ;

         PROPERTY ioCellOriginX 25.0 ;

         PROPERTY ioCellOriginY 0.0 ;

     END PAD

 END IOCELL
```

```
MACRO BUMPA
     CLASS COVER BUMP
     ...
     PROPERTY ioCellOffsetX -40 ;
     PROPERTY ioCellOffsetY -70 ;
 END BUMPA
```

Innovus automatically names the stagger bond pads as:

*IoCellInstName*_PAD*#*

Here:

- *IoCellInstName* - Specifies the instance name of the I/O cell on which the bond pad is being placed.
- *#* - Specifies the incremented bond pad number. The number `1` is used for all single pad I/O cells. This number is incremented when there are multiple bond pads on a single I/O cell.
  For example, if I/O cell `CKdiff` contained two pins, two bond pads would be placed on this cell and named as `CKdiff_PAD1` and `CKdiff_PAD2`.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `place_bond_pad_by_stagger` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man place_bond_pad_by_stagger`
`-all`	Specifies that bond pads will be placed on all I/O instances. *Data_type*: bool, optional
`-bond_pad` *`position padName position padName...`*	
	Specifies which bond pad and position to use when multiple bond pads are specified in the LEF file for a given inner (i), middle (m), or outer (o) position. For example, If there are three outer bond pads specified in the LEF file, you can use this parameter to choose which bond pad to use as the outer bond pad. *Default*: The software uses the first bond pad in the list of bond pads for a given position. *Data_type*: string, optional
`-first_io_inst_name` *`instName`*	
	Specifies the name of the first I/O instance of where bond pads will be placed. You can use this parameter in conjunction with the `-last_io_inst` parameter. If multiple pins exist in the instance, you can use the `-pin` parameter to specify which pin to start with. **Note:** Bond pads are placed in a clock-wise pattern from the starting I/O instance (`-first_io_inst_name`) to the last I/O instance (`-last_io_inst`) if applicable. *Data_type*: string, optional
`-last_io_inst` *`instName`*	Specifies the name of the last I/O instance of where bond pads will be placed. You can use this parameter in conjunction with the `-first_io_inst_name` parameter. **Note:** Bond pads are placed in a clock-wise pattern from the starting I/O instance (`-first_io_inst_name`) to the last I/O instance (`-last_io_inst`). *Data_type*: string, optional
`-pattern` *`pattern_string`*	Specifies the stagger pattern of the bond pads on the I/O instances. *`pattern_string`* specifies any combination of `i` (inner), `m` (middle), and `o` (outer). *Data_type*: string, optional
`-pin` *`pinName`*	Specifies the name of the pin inside the instance. If the instance specified by the `-first_io_inst_name` parameter contains multiple pins, you use this parameter to specify which pin to start with. *Data_type*: string, optional
`-ring` *`number`*	Specifies which ring to operate on with ring 1 being the outer most ring. The ring number is incremented by one as you get closer to the core. For a three-ring design, the inner most ring is ring 3. *Data_type*: int, optional
`-side {east west north south}`	
	Specifies the geographical side of the die. You can specify the `east`, `west`, `north`, or `south` side of the die. *Data_type*: enum, optional
`-start_stagger_position {inner middle outer}`	
	Specifies the starting stagger position of the specified bond pad (`-bond_pad`) on the specified I/O instance (`-first_io_inst_name`). The default stagger position sequence is `inner`, `middle`, and `outer`. If you start with a stagger position of `middle`, the next bond pad is placed in an `outer` stagger position followed by a bond pad placed in an `inner` stagger position. *Data_type*: enum, optional

Examples

- The following command places bond pads on all I/O instances using a stagger pattern of inner, outer, and middle and using bond pad `BUMPA1` at the outer position:
  ```
  place_bond_pad_by_stagger -all -pattern iom -bond_pad o BUMPA1
  ```
- The following command places bond pads on the I/O instances of the outer most ring (East side) starting with a bond pad placed in the middle stagger position:
  ```
  place_bond_pad_by_stagger -side east -start_stagger_position middle -ring 1
  ```
- The following command places bond pads from I/O instance `abc` to I/O instance `xyz`, starting with pin `PAD` of I/O instance `abc` and using an inner, inner, outer stagger pattern:
  ```
  place_bond_pad_by_stagger -first_io_inst_name abc -last_io_inst xyz -pin PAD -pattern iio
  ```

Related Information

- `check_bond_pad_spacing`
- `get_bond_pad`
- `place_bond_pad`
- `reset_fixed_bond_pad`
- `set_bond_pads_to_fixed`
- `space_bond_pads`

read_bump_color_map_file

```
read_bump_color_map_file
[-help]
bump_color_file
```

Loads the bump color map file to change the color of assigned bumps. The command supports all the colors specifed in the Qt color table. A list of recognized color names can be found at `http://www.w3.org/TR/SVG/types.html#ColorKeywords`.

The bump color map file supports wildcards for ease of use. For example, instead of specifying `tdigit[1] green`, `tdigit[2] green` and so on, you can specify `tdigit* green` in the bump color map file to color all `tdigit` bumps green.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `read_bump_color_map_file` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man read_bump_color_map_file`
`bump_color_file`	Specifies the name of the bump color map file to load. *Data_type*: string, required

Example

- The following command loads the bump color map file `BumpColor.Map` and assigns colors to bumps in the design:
  ```
  read_bump_color_map_file BumpColor.Map
  ```
  Output:
  ```
  port_pad_data_out[8] wheat
  tdigit* green
  ```

Related Information

- `assign_bumps`

read_flip_chip_property

```
read_flip_chip_property
[-help]
in_file
[-only_bump_connect_target_constraint]
```

Reads the text file containing flip chip property information, including virtual connection area property. After the file is loaded, the virtual connection area property is added to IO pads and can be displayed and used for flip chip flow. The `read_flip_chip_property` command prints an error message if the following errors occur:

- Type error of property name
- Instance/pin not found
- Metal layer not found
- Size is zero or minus

Parameters

`-help`	Prints out the command usage that includes type and default information for each `read_flip_chip_property` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man read_flip_chip_property`
in_file	Specifies the bump and IO pad property text file, containing virtual connection area property information. *Data_type*: string, required
`-only_bump_connect_target_constraint`	Reads only the bump connect target constraints. *Data_type*: bool, optional

Example

The following command reads only the bump connect target constraints from the `test2` input file:

```
read_flip_chip_property test2 -only_bump_connect_target_constraint
```

Related Information

- `read_io_update`
- `write_flip_chip_property`
- Flip Chip Methodologies chapter of the *User Guide*
- Tools Menu chapter in the *Menu Reference*
  - *Flip Chip*

read_io_update

```
read_io_update
[-help]
[-check_only]
in_file
```

Reads a simple ASCII file which contains the bump and instance information.

The input die abstract file is a simple ASCII file that consists of bump and I/O instance information only; these are the only two objects that can be modified by SiP Layout.

- If a bump is defined in the db and not defined in the input file, the bump gets deleted from the db.
- If the instance names in the input file and db do not match, an error is displayed.
- If the nets referenced in the bump section in the input file do not exist in the db, an error is displayed.

Parameters

`-help`	Prints out the command usage that includes type and default information for each `read_io_update` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man read_io_update`
`-check_only`	Reports differences between the input file and the design in memory. This is an optional parameter. This parameter only generates a report and does not update the die. *Data_type*: bool, optional
in_file	Specifies a file containing the state of I/O drivers and bumps (placement and assignments). This is a required parameter. The information from this file is updated in the database. *Data_type*: string, required

Related Information

- `read_flip_chip_property`
- `read_package`
- `write_flip_chip_property`
- Flip Chip Methodologies chapter in the *User Guide*
- Tools Menu chapter in the *Menu Reference*
  - *Flip Chip*

read_package

```
read_package
[-help]
package_file
```

Reads the package file dumped out by the System-in-Package (SiP) layout in XML format and displays the package ball/finger location in the Innovus floorplan view. The command also displays the flight lines from the IO pads to fingers for a wire-bond design or from bumps to package balls for a flip chip design.

The package overlay schema dumped out by SiP includes bond finger, package balls, and many other package objects. `read_package` displays all the package objects for a wire-bond or flip chip design, such as BGA_BALL, BOND_FINGER, BONDWIRE and ROUTE, in the Innovus main window. The exact information displayed depends on the package overlay file dumped out by SiP. You can turn on or turn off the display of specific package objects using the Custom tab of the Color Preferences form in Innovus. You can view attributes of a selected package object using Attribute Editor.

Parameters

`-help`	Prints out the command usage that includes type and default information for each `read_package` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man read_package`
`package_file`	Specifies the name of the package file. *Data_type*: string, required

report_probe_pins

```
report_probe_pins [-help]
```

Generates a report of all probe pins that are used for package-level design and testing. If no probe points exist, the software returns the following message:

```
"No Probe Points Defined"
```

Note: You can set or delete probe points using the `set_probe_pin` or `reset_probe_pin` commands, respectively.

Parameters

`-help`	Prints a brief description of the `report_probe_pins` commabd. For a detailed description of the command, use the `man` command: `man report_probe_pins`

Example

- The following command generates a report of all probe pins:
  `report_probe_pins`

Related Information

- `reset_probe_pin`
- `set_probe_pin`

report_special_routes

```
report_special_routes
[-help]
[-nets netName]
[-center_line]
[-out_file report_name]
[[ [-by_layer]
   [-highlight_45_layer layerName]
   [-length_threshold value]
   [-resistance_file resFileName]
   [-selected]
   [-summary]
   [-wire_length]
   [-wire_segment]]
 | [-interposer_route
   [[-exclude_nets netName]
    [-interposer_layers layerName]
    [-interposer_route_constraint_file ircFileName]]]]
```

Reports the length of all RDL routes (SPECIALNETS) created with the `route_flip_chip` command. This includes support for Manhattan and 45-degree routes. You can report the wire length of each SPECIALNET and report the SPECIALNETS that are greater in length than the specified threshold value.

Use the `-by_layer` and `-wire_segment` options to report the width and length of the special nets per layer and wire segment.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `report_special_routes` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man report_special_routes`

`-by_layer`	Reports on layers of each special net. Here's a sample format of the report when the `-by_layer` option is specified:
	`NET: port_pad_data_in[0] TOTAL SEGMENTS 4 WIRE LENGTH 303.4980 VIAS 1`
	`LAYER 8(METAL8) TOTAL LENGTH 303.4980`
	`IOWIRE SEGMENTS 4 WIRE LENGTH 303.4980 VIAS 1`
	Data_type: bool, optional

`-center_line`	Specifies that routing length is calculated by center line. That is, the sum of wire center line is used to calculate wire length. Without this parameter, the routing length is calculated as sum of wire segments. *Data_type*: bool, optional
`-exclude_nets netName`	
	Specifies the net names to be excluded. This option can be used only if `-interposer_route` is specified. *Data_type*: string, optional
`-highlight_45_layer layerName`	
	Highlights (in yellow) all nets with 45-degree routing on the specified layer in the design display area. *Data_type*: string, optional
`-interposer_layers layerName`	
	Specifies the interposer layers. Use this option to report the maximum and minimum routing length from tap via to tap via and from bum This option can be used only if `-interposer_route` is specified. *Data_type*: string, optional

`-interposer_route`	Reports the interposer route statistics, including maximum, minimum, and average wire length of the interposer routing. `report_special` `set_integration_route_constraint` to decide which nets are in one group. The `set_integration_route_constraint` constraints are e by using the `-interposer_route_constraint_file` option.
	The report format is as shown below:
	`### Header ###`
	`### Total Nets of Interposer Route ###`
	`Total nets: <total nets number>`
	`Layer: Layer_name`
	`Net with Max wire length: <net name + net length>`
	`Net with Min wire length: <net name + net length>`
	`Average wire length:`
	
	`### Net Group <group_name1> ###`
	`Total nets: <nets number in group>`
	`Layer: Layer_name`
	`Net with Max wire length:`
	`<net name> :`
	`net length :`
	`Euclidean distance:`
	`Manhattan distance:`
	`Net with Min wire length:`
	`<net name> :`
	`net length :`
	`Euclidean distance:`
	`Manhattan distance:`
	`Average wire length:`
	`Nets name:`
	
	`### Net Group <group_name2> ###`
	`Total nets: <nets number in group>`
	`Layer: Layer_name`
	`Net with Max length: <net name + net length>`
	`...`
	`Net with Min length: <net name + net length>`
	`...`
	`Average wire length:`
	`Nets name:`
	`...`
	The *`group_name`* of the nets is obtained from `set_integration_route_constraint -name` *`group_name`*. If `-name` is not specified, the tool us
	For example: `### Net Group <1> ###`

`-interposer_route_constraint_file` *`irc_file`*	
	Specifies the name of the interposer route constraint file. This option can be used only if `-interposer_route` is specified.
`-length_threshold` *`value`*	

	Reports on the SPECIALNETS that are greater in length than the specified value. These values are listed in the output file specified by -ou *Data_type*: float, optional
-nets netName	Reports on the wire length of each SPECIALNET fora specified net. **Note:** You must use this parameter in conjunction with the -wire_length or -length_threshold parameters. *Data_type*: string, optional
-out_file *report_name*	
	Specifies the report file name. *Data_type*: string, optional
-resistance_file resFileName	
	Specifies the resistance file name (fcroute.res) generated by fcroute if srouteFCReport *file name* is specified in the config file and h not match the max resistance constraint defined in the fcroute constraint file. *Data_type*: string, optional
-selected	Reports the length of the selected wire segments for each special net. *Data_type*: bool, optional
-summary	Summarizes the wire length and VIAs for special nets. Here's a sample format of the report when -summary is specified: *Data_type*: bool, optional

```
-----------------------------------------------------------------

-----------------------------------------------------------------
Report wire length and VIAs per layer
-----------------------------------------------------------------
 8(METAL8)  Total wire segments 15 Total length 1243.527000
  ------------------------------------------------
    #Shape           Wire segment    Length
     Horizontal      5                 297.6950
     Vertical        5                 466.4500
     45degree        5                 479.3820
  ------------------------------------------------
cutlayer VIA78 Total VIAs: 4 Total Cuts: 1360

-----------------------------------------------------------------

-----------------------------------------------------------------
Report wire length per shape for all layers
------------------------------------------------
 #Shape           Wire segment    Length
  Horizontal      5                 297.6950
  Vertical        5                 466.4500
  45degree        5                 479.3820
-----------------------------------------------------------------


Total Wire Length of All Special Nets: 1243.5270
```

`-wire_segment`	Reports on the wire segments of each special net. Here's a sample format of the report when `-wire_segment` is specified:
	
	`NET: port_pad_data_in[1] TOTAL SEGMENTS 4 WIRE LENGTH 251.9120 VIAS 1`
	`WIRESEGMENT 1 SHAPE Vertical LENGTH 110.7050 674.7200 880.7200 686.7200 991.4250`
	`WIRESEGMENT 2 SHAPE 45degree LENGTH 76.3320 620.7450 985.4250 686.7200 1051.4000`
	`WIRESEGMENT 3 SHAPE Horizontal LENGTH 38.3550 588.3900 1039.4000 626.7450 1051.4000`
	`WIRESEGMENT 4 SHAPE Vertical LENGTH 26.5200 588.3900 1039.4000 597.3900 1065.9200`
	`IOWIRE SEGMENTS 4 WIRE LENGTH 251.9120 VIAS 1`
	
	Data_type: bool, optional

`-wire_length`	Reports on the wire length of each `SPECIALNET`. *Data_type*: bool, optional

Example

- The following command reports the wire length of each `SPECIALNET` and lists it in the `spwire.rpt` report file:
  `report_special_routes -out_file spwire.rpt -wire_length`

  Example report file:

```
############################################################
# Generated by: Cadence Innovus 15.10-b024_1
# OS: Linux x86_64(Host ID icdcmopt03s)
# Generated on: Wed Jul 17 15:32:48 2013
# Design: DTMF_CHIP
# Command: report_special_routes -out_file spwire.rpt -wire_length
############################################################

NET: VDD TOTAL SEGMENTS 0 WIRE LENGTH 0.0000 VIAS 0

NET: VSS TOTAL SEGMENTS 0 WIRE LENGTH 0.0000 VIAS 0

NET: ibias TOTAL SEGMENTS 0 WIRE LENGTH 0.0000 VIAS 0

NET: int TOTAL SEGMENTS 0 WIRE LENGTH 0.0000 VIAS 0

NET: pllrst TOTAL SEGMENTS 0 WIRE LENGTH 0.0000 VIAS 0

NET: port_pad_data_in[0] TOTAL SEGMENTS 4 WIRE LENGTH 303.4980 VIAS 1
WIDTH 12.0000 LENGTH 303.4980
IOWIRE SEGMENTS 4 WIRE LENGTH 303.4980 VIAS 1

NET: port_pad_data_in[10] TOTAL SEGMENTS 0 WIRE LENGTH 0.0000 VIAS 0

NET: port_pad_data_in[11] TOTAL SEGMENTS 0 WIRE LENGTH 0.0000 VIAS 0

NET: port_pad_data_in[12] TOTAL SEGMENTS 0 WIRE LENGTH 0.0000 VIAS 0

NET: port_pad_data_in[13] TOTAL SEGMENTS 0 WIRE LENGTH 0.0000 VIAS 0

NET: port_pad_data_in[14] TOTAL SEGMENTS 0 WIRE LENGTH 0.0000 VIAS 0

NET: port_pad_data_in[15] TOTAL SEGMENTS 0 WIRE LENGTH 0.0000 VIAS 0

NET: port_pad_data_in[1] TOTAL SEGMENTS 4 WIRE LENGTH 251.9120 VIAS 1
WIDTH 12.0000 LENGTH 225.3920
WIDTH 9.0000 LENGTH 26.5200
IOWIRE SEGMENTS 4 WIRE LENGTH 251.9120 VIAS 1

NET: port_pad_data_in[2] TOTAL SEGMENTS 0 WIRE LENGTH 0.0000 VIAS 0

NET: port_pad_data_in[3] TOTAL SEGMENTS 0 WIRE LENGTH 0.0000 VIAS 0

NET: port_pad_data_in[4] TOTAL SEGMENTS 0 WIRE LENGTH 0.0000 VIAS 0

NET: port_pad_data_in[5] TOTAL SEGMENTS 0 WIRE LENGTH 0.0000 VIAS 0

NET: port_pad_data_in[6] TOTAL SEGMENTS 0 WIRE LENGTH 0.0000 VIAS 0

NET: port_pad_data_in[7] TOTAL SEGMENTS 0 WIRE LENGTH 0.0000 VIAS 0

NET: port_pad_data_in[8] TOTAL SEGMENTS 0 WIRE LENGTH 0.0000 VIAS 0

NET: port_pad_data_in[9] TOTAL SEGMENTS 0 WIRE LENGTH 0.0000 VIAS 0

NET: port_pad_data_out[0] TOTAL SEGMENTS 0 WIRE LENGTH 0.0000 VIAS 0

NET: port_pad_data_out[10] TOTAL SEGMENTS 0 WIRE LENGTH 0.0000 VIAS 0
```

```
NET: port_pad_data_out[11] TOTAL SEGMENTS 0 WIRE LENGTH 0.0000 VIAS 0

NET: port_pad_data_out[12] TOTAL SEGMENTS 0 WIRE LENGTH 0.0000 VIAS 0

NET: port_pad_data_out[13] TOTAL SEGMENTS 0 WIRE LENGTH 0.0000 VIAS 0

NET: port_pad_data_out[14] TOTAL SEGMENTS 0 WIRE LENGTH 0.0000 VIAS 0

NET: port_pad_data_out[15] TOTAL SEGMENTS 0 WIRE LENGTH 0.0000 VIAS 0

NET: port_pad_data_out[1] TOTAL SEGMENTS 0 WIRE LENGTH 0.0000 VIAS 0

NET: port_pad_data_out[2] TOTAL SEGMENTS 0 WIRE LENGTH 0.0000 VIAS 0

NET: port_pad_data_out[3] TOTAL SEGMENTS 0 WIRE LENGTH 0.0000 VIAS 0

NET: port_pad_data_out[4] TOTAL SEGMENTS 0 WIRE LENGTH 0.0000 VIAS 0

NET: port_pad_data_out[5] TOTAL SEGMENTS 0 WIRE LENGTH 0.0000 VIAS 0

NET: port_pad_data_out[6] TOTAL SEGMENTS 0 WIRE LENGTH 0.0000 VIAS 0

NET: port_pad_data_out[7] TOTAL SEGMENTS 0 WIRE LENGTH 0.0000 VIAS 0

NET: port_pad_data_out[8] TOTAL SEGMENTS 3 WIRE LENGTH 435.8845 VIAS 1
WIDTH 9.0000 LENGTH 166.5900
WIDTH 12.0000 LENGTH 269.2945
IOWIRE SEGMENTS 3 WIRE LENGTH 435.8845 VIAS 1

NET: port_pad_data_out[9] TOTAL SEGMENTS 0 WIRE LENGTH 0.0000 VIAS 0

NET: refclk TOTAL SEGMENTS 0 WIRE LENGTH 0.0000 VIAS 0

NET: reset TOTAL SEGMENTS 0 WIRE LENGTH 0.0000 VIAS 0

NET: scan_clk TOTAL SEGMENTS 0 WIRE LENGTH 0.0000 VIAS 0

NET: scan_en TOTAL SEGMENTS 0 WIRE LENGTH 0.0000 VIAS 0

NET: scan_in_1 TOTAL SEGMENTS 0 WIRE LENGTH 0.0000 VIAS 0

NET: scan_in_2 TOTAL SEGMENTS 0 WIRE LENGTH 0.0000 VIAS 0

NET: scan_out_1 TOTAL SEGMENTS 0 WIRE LENGTH 0.0000 VIAS 0

NET: scan_out_2 TOTAL SEGMENTS 0 WIRE LENGTH 0.0000 VIAS 0

NET: spi_data TOTAL SEGMENTS 0 WIRE LENGTH 0.0000 VIAS 0

NET: spi_fs TOTAL SEGMENTS 0 WIRE LENGTH 0.0000 VIAS 0

NET: tdigit[0] TOTAL SEGMENTS 0 WIRE LENGTH 0.0000 VIAS 0

NET: tdigit[1] TOTAL SEGMENTS 0 WIRE LENGTH 0.0000 VIAS 0

NET: tdigit[2] TOTAL SEGMENTS 0 WIRE LENGTH 0.0000 VIAS 0

NET: tdigit[3] TOTAL SEGMENTS 0 WIRE LENGTH 0.0000 VIAS 0

NET: tdigit[4] TOTAL SEGMENTS 0 WIRE LENGTH 0.0000 VIAS 0

NET: tdigit[5] TOTAL SEGMENTS 4 WIRE LENGTH 252.2325 VIAS 1
WIDTH 12.0000 LENGTH 213.2325
WIDTH 9.0000 LENGTH 39.0000
IOWIRE SEGMENTS 4 WIRE LENGTH 252.2325 VIAS 1

NET: tdigit[6] TOTAL SEGMENTS 0 WIRE LENGTH 0.0000 VIAS 0

NET: tdigit[7] TOTAL SEGMENTS 0 WIRE LENGTH 0.0000 VIAS 0

NET: tdigit_flag TOTAL SEGMENTS 0 WIRE LENGTH 0.0000 VIAS 0

NET: test_mode TOTAL SEGMENTS 0 WIRE LENGTH 0.0000 VIAS 0

NET: vcom TOTAL SEGMENTS 0 WIRE LENGTH 0.0000 VIAS 0

NET: vcop TOTAL SEGMENTS 0 WIRE LENGTH 0.0000 VIAS 0


Total Wire Length of All Special Nets: 1243.5270
```

Related Information

- report_wires
- route_flip_chip

reset_bumps_fixed_assign

```
reset_bumps_fixed_assign
[-help]
{-all | -bumps {bump1 bump2...}}
```

Removes the bump assignment status from a specified list of assigned bumps that were set with the `set_bumps_fixed_assign` command. The software updates the DEF file by removing the `+FIXEDBUMP` attribute from the specified group of assigned bumps to indicate that they are now in an unfixed state.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `reset_bumps_fixed_assign` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man reset_bumps_fixed_assign`
`-all`	Removes the bump assignment status from all bumps. *Data_type*: bool, required
`-bumps {bump1 bump2...}`	Removes the bump assignment status from a specified list of assigned bumps. *Data_type*: string, required

Examples

- The following command removes the bump assignment status from all bumps:
  `reset_bumps_fixed_assign -all`
- The following command removes the bump assignment status from bumps `bumpA` and `bumpB`:
  `reset_bumps_fixed_assign -bumps {bumpA bumpB}`

Related Information

- `assign_bumps`
- `set_bumps_fixed_assign`
- `write_def`
- Flip Chip Methodologies chapter in the *User Guide*
- Tools Menu chapter in the *Menu Reference*
  - *Flip Chip*

reset_fixed_bond_pad

```
reset_fixed_bond_pad
[-help]
{{-io_inst InstName [-pin pinName]} | -selected}
```

Unassigns the fixed status of a specified bond pad.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `reset_fixed_bond_pad` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man reset_fixed_bond_pad`
`-io_inst instName`	Specifies the name of the I/O instance whose bond pad's fixed status will be unassigned. *Data_type*: string, required
`-pin pinName`	Specifies the name of the pin. *Data_type*: string, optional
`-selected`	Unassigns the fixed status of a selected bond pad. You must first select the bond pad in the design display area, then use this parameter to unassign the fixed status. *Data_type*: bool, required

Example

- The following command unassigns the fixed status of the bond pad on I/O instance `pad_addr[30]`:
  ```
  reset_fixed_bond_pad -io_inst {pad_addr[30]}
  ```

Related Information

- `check_bond_pad_spacing`
- `get_bond_pad`
- `place_bond_pad`
- `place_bond_pad_by_stagger`
- `set_bond_pads_to_fixed`
- `space_bond_pads`

reset_probe_pin

```
reset_probe_pin
[-help]
{{-inst InstName -pin pinName} | -bump bumpName }
```

Unsets a pad or bump as a probe point. Probe points are used for package-level design and testing. To set or report on probe points, use the `set_probe_pin` or `report_probe_pins` commands, respectively.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `reset_probe_pin` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man reset_probe_pin`
`-bump bumpName`	Specifies the instance name of the bump. *Data_type*: string, required
`-inst InstName`	Specifies the name of the I/O instance. *Data_type*: string, required
`-pin pinName`	Specifies the name of the pin on the I/O instance that will be unset as a probe point. *Data_type*: string, required

Example

- The following command unsets the `PAD` pin of `Bump_1` as a probe point:
  ```
  reset_probe_pin -inst Bump_1 -pin PAD
  ```

Related Information

- `report_probe_pins`
- `set_probe_pin`

route_flip_chip

```
route_flip_chip
[-help]
[-bottom_layer layerName]
[-connect_in_region]
[-connect_tsv_to_bump] | [-connect_tsv_to_pad] | [-connect_tsv_to_ring_stripe]
[-constraint_file filename]
[-delete_existing_routes]
[-double_bend_route]
[-eco | -incremental]
[-extra_config fileName]
[-jog_control {no_jog | jog_on_the_same_layer | jog_on_different_layers}]
[-keep_drc]
[-min_escape_distance unit]
[-msg_rate int]
[-nets {net_name_list | <filename | ~<filename} | -selected_bumps]
[-only_global]
[-overflow_map]
[-region {x1 y1 x2 y2}]
[-route_engine {detail global_detail}]
[-route_pg_style {none finger mesh}]
[-route_width real]
[-spread_wires_factor value]
[-straight_connections {straightwithdrcclean | straightwithchanges}]
-target {connect_bump_to_ring_stripe | connect_bump_to_pad}
[-top_layer layerName]
[-user_class subclass_string]
[-verbose]
```

Specifies that power routing and signal routing recognize the bumps specified in a flip chip design.

- The command supports all the `LEF 5.7` syntax to determine which power/ground pin shape on the I/O driver cell must be connected to a bump. For more information, see the "Performing Area I/O Placement" section in the Data Preparation chapter of the *Innovus Stylus Common UI User Guide*.
- The command supports mixed-mode designs in the detail and global_detail modes.
- The command, by default, reads the FE database to get all information. The default value of the variable, `set rda_global (srouteUseFEDB)` must be set to `1`. For the command to read information from the LEF file, the `set rda_global (srouteUseFEDB)` variable must be set to `0`.

Use this command after importing the design.

You can perform connect_bump_to_pad routing (`-target connect_bump_to_pad`) using the detail or global_detail routing styles. Each routing style supports a specific set of `route_flip_chip` command parameters. The following table lists the routing style support for each parameter.

Note: The following table might not be readable when viewed with the Unix `man` command. Check the `route_flip_chip` command description in the *Text Command Reference* to view the table.

route_flip_chip Parameters	-route_engine	
	detail	**global_detail**
-region {x1 y1 x2 y2}	X	X1
-connect_in_region	X	
-connect_tsv_to_bump	X	
-connect_tsv_to_pad	X	
-connect_tsv_to_ring_stripe	NA	NA
-constraint_file filename	X	X
-delete_existing_routes	X	X
-double_bend_route	X	X
-eco	X	X
-extra_config fileName	X	X

`-only_global`		X
`-incremental`		X
`-jog_control {no_jog | jog_on_the_same_layer | jog_on_different_layers}`	X	
`-keep_drc`	X	X
`-bottom_layer` *layerName*	X	X
`-top_layer` *layerName*	X	X
`-min_escape_distance` *unit*	X	X
`-msg_rate` *int*	X	X
`-nets {`*net_name_list* `| <filename | ~<filename}`	X	X
`-overflow_map`		X
`-route_width` *real*	X	X
`-selected_bumps`	X	X
`-spread_wires_factor` *value*	X	X
`-straight_connections {straightwithdrcclean | straightwithchanges}`	NA	NA
`-target {connect_bump_to_ring_stripe | connect_bump_to_pad}`	NA	NA
`-verbose`	X	X

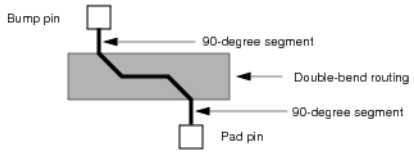
X[1]	When run incrementally, the `route_flip_chip` global_detail mode partially supports area-based routing in the specified coordinates (`route_flip_chip -region {`*x1 y1 x2 y2*`} -incremental`) for the area in which the net is routed.

Parameters

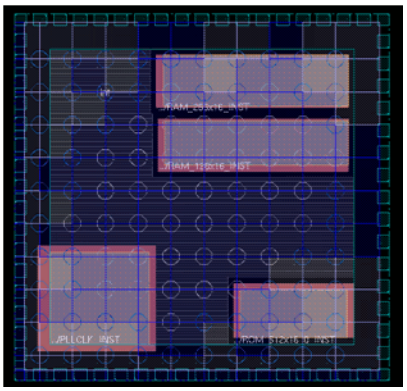
`-help`	Outputs a brief description that includes type and default information for each `route_flip_chip` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man route_flip_chip`

`-bottom_layer` *layerName*	Specifies the bottom-most metal layer number / `LEF` layer names that the software can use when routing bumps.
	Default: If you do not specify this parameter, all layers are used.
	For example, you can specify the metal layer numbers, `8` and `2`, as follows:
	`route_flip_chip -target connect_bump_to_ring_stripe -route_width 4.0 -top_layer 8 -bottom_layer 2`
	You can also specify the `LEF` layer names, `METAL7` and `METAL1`, as follows:
	`route_flip_chip -target connect_bump_to_ring_stripe -route_width 4.0 -top_layer METAL7 -bottom_layer METAL1`
	Data_type: string, optional

`-connect_in_region`	Specifies that connections from all sources within the specified area can connect only to targets that are also inside that area. **Note:** This parameter is only available if you also specify the `-region` parameter. *Default:* If you do not specify this option the software makes connections from all sources within the specified area to targets both inside and outside the specified area. *Data_type*: bool, optional

`-connect_tsv_to_bump`	Connects TSVs to signal bumps. Use the `-connect_tsv_to_bump` parameter with `-target connect_tsv_to_pad`. This parameter cannot be used along with `-connect_tsv_to_pad` and `-connect_tsv_to_ring_stripe`, which provide other routing options. **Note:** Different layers have different types of bumps. It is recommended that you route the bump on the back side and front side separately using the `srouteExcludeBumpType` statement in the extra configuration file. *Data_type*: bool, optional
`-connect_tsv_to_pad`	Connects TSVs to IO pads. Use the `-connect_tsv_to_pad` parameter with `-target connect_bump_to_pad`. *Data_type*: bool, optional
`-connect_tsv_to_ring_stripe`	Connects TSV with power net to the power mesh. Use the `-connect_tsv_to_ring_stripe` parameter with `-target connect_bump_to_ring_stripe`. For more information, see "TSV/Bump/Back Side Metal Modeling in Innovus" section of the Design Methodology for 3D IC with Through Silicon Via chapter in the *Innovus Stylus Common UI User Guide*. *Data_type*: bool, optional
`-constraint_file` *`filename`*	Specifies the file that contains constraints for flip chip routing. For more information, see "Useful Constraints for Flip Chip Routing" in the Flip Chip Methodologies chapter of the *Innovus Stylus Common UI User Guide.* *Data_type*: string, optional
`-delete_existing_routes`	Specifies that the software will remove existing connections when you use the `route_flip_chip` command multiple times. *Default:* The software maintains existing connections each time you issue the `route_flip_chip` command. **Note:** If you specify the `-region` parameter, existing routes are always preserved, even if you also specify the `-delete_existing_routes` parameter. *Data_type*: bool, optional
`-double_bend_route`	Allows double-bend routing on segments between a bump pin and a pad pin.  **Note:** Segments that connect directly to a bump pin or pad pin are always routed at 90 degrees. *Default*: Off *Data_type*: bool, optional
`-eco`	Enables ECO mode flip chip routing. When you specify this parameter, `route_flip_chip` automatically performs typical ECO routing steps, such as deleting existing routing results or re-routing affected nets, whenever there is an ECO change. For more information, see the "ECO Routing" section in the Flip Chip Methodologies chapter of the *Innovus Stylus Common UI User Guide*. **Note:** The `-eco` parameter cannot be used along with `-incremental`. *Data_type*: bool, optional
`-extra_config` *`filename`*	Specifies the name of an extra configuration file. This file contains additional route_flip_chip commands that provide more control over the routes to bumps. For example, `route_flip_chip -target connect_bump_to_pad -route_engine global_detail -extra_config mixmode.config` For more information, see "Useful Extra Configurations for Flip Chip Routing" in the Flip Chip Methodologies chapter of the *Innovus Stylus Common UI User Guide.* *Data_type*: string, optional

`-incremental`	Runs the global_detail mode incrementally when pre-routes are already in the design, without deleting or re-routing the pre-routes. **Note:** The `-incremental` parameter cannot be used along with `-eco`. *Data_type*: bool, optional
`-only_global`	Executes the global router to provide you with the routability statistics. This is helpful for optimizing mode. *Data_type*: bool, optional
`-jog_control {no_jog | jog_on_the_same_layer | jog_on_different_layers}`	
	Specifies that jogs are allowed during routing to avoid DRC violations. You must specify either the `-jog_control` parameter or the `-straight_connections` parameter. If you specify the `-jog_control` parameter, you can also specify one of the following options: • `no_jog`: Specify this option if you prefer the software to make straight connections between targets, and to change layers instead of jog to avoid DRC violations. If you do not specify this option, the route uses both layer changes and jogging to avoid DRC violations. • `jog_on_the_same_layer`: If the route must jog to avoid a DRC violation, the jog occurs on the same layer whenever possible. This can result in routing in the non-preferred direction. • `jog_on_different_layers`: If the route must jog to avoid a DRC violation, the jog occurs on the layer that is in the preferred routing direction whenever possible. *Data_type*: (enum)+, optional
`-keep_drc`	If specified, the command `delete_drc_markers` will not run implicitly when executing the `route_flip_chip` command, hence retaining the DRC violations. Otherwise, the command `clearDrc` runs implicitly when executing the `route_flip_chip` command by default, purging all the violation markers. *Data_type*: bool, optional
`-min_escape_distance` *value*	Specifies the minimum distance from the edge of the pin to the edge of the wire segment. *Data_type*: float, optional
`-msg_rate` *int*	Specifies the interval at which progress messages are generated. If you specify `1`, the software generates every progress message; if you specify `2`, the software generates every other progress message, and so on. *Default:* `0` *Data_type*: int, optional
`-nets {`*net_name_list* `| <filename | ~<filename}`	
	Specifies the nets to be connected or excluded from connection. *Default:* If you do not specify the `-nets` parameter, the software connects all power and ground nets in the design. Use one of the following options: • *net_name_list*: Specifies names of the nets to be connected. To specify more than one net, enclose the net names in braces and separate the names with a space. **Note:** As the `-nets` option accepts a maximum of 4096 string characters directly, use the *net_name_list* option if the number of nets to be connected is not very high. If a large number of nets need to be connected, use the `<filename` option or `~<filename` option with `-nets`. • `<filename`: Specifies the name of a text file containing the list of nets to be connected. Use this option if the number of nets to be connected is high. In the text file, separate net names with commas or the newline, space, or tab character. Wildcards can be used when specifying net names in the text file. • `~<filename`: Excludes the nets in the specified text file during flip chip routing. Use this option if the number of nets to be excluded is much lower than the number of nets to be connected. Wildcards can be used when specifying net names in the text file. *Data_type*: string, optional

`-overflow_map`	Displays the thermal map using the width and spacing parameters specified in the extra configuration file. To display the thermal map in the floorplan view in the GUI, select *All Colors* which opens the Color Preferences form, then select *GCell Overflow*. **Note:** The routing congestion can also be displayed using the `set_layer_preference` command. (`set_layer_preference gcellOvflow -isVisible 1`)  *Data_type*: bool, optional
`-region {x1 y1 x2 y2}`	Specifies the coordinates for detail routing. You must enclose the x and y coordinates in curly braces and separate them with a space. **Notes:** • Innovus issues a warning message if you specify area constraints in the flip chip route global_detail mode. (`route_flip_chip -route_engine global_detail`) • If `-route_pg_style` is specified as `finger`, the PG finger is routed in the area specified with `-region`. If `-route_pg_style` is not specified or specified as `none`, area I/O routing is done within the coordinates specified with `-region`. *Data_type*: rect, optional
`-route_engine {detail | global_detail}`	
	Selects between detail and global_detail routing styles. The `detail` mode supports area-based routing (`route_flip_chip -region`). However, the `global_detail` mode partially supports area-based routing only if you set the `-incremental` parameter. You can run `route_flip_chip` in the `global_detail/detail` mode when pre-routes are already in the design, without deleting or re-routing the pre-routes. By default, this feature is true for `route_flip_chip -target connect_bump_to_pad -route_engine detail`. For the `global_detail` mode, you should add the `-incremental` parameter to enable this feature. *Default*: `global_detail` *Data_type*: enum, optional
`-route_pg_style {none finger mesh}`	
	Specifies the routing pattern for flip chip route power routing: • `none`: For normal PG routing • `finger`: For PG finger routing style, in which you need to specify the routing direction. This option must be used with `route_flip_chip -region` option. PG finger routing improves usability and provides better IR drop by lowering rail resistance. • `mesh`: For mesh-style PG routing on the redistribution layer (RDL). This option is used to increase PG density. If you specify PG mesh routing, you must specify the main width of the PG mesh by using the `flip_chip_pg_mesh_main_width` attribute. *Default:* `none` *Data_type*: enum, optional

`-route_width` *real*	Connects wires to bumps using the specified width only, regardless of the size of the bump or the source to which the bump connects. *Default:* When using the `global_detail` routing style, `route_flip_chip` determines the width of the route as the bump width or I/O cell pin width that is connected to the bump, whichever is smaller. When using the detail routing style, `route_flip_chip` does not restrict the size of the RDL route to the width of the I/O cell pin and can route a wire that is larger in width than the specified I/O cell pin width. To override the above default setting for detail routing style, you can use the `srouteUseSpecifiedWidthForTopLayer TRUE` and `srouteRouteWidthForLowerLayer` *db_unit* variables in the extra configuration file. This forces `route_flip_chip` to use user-defined width for the top layer connecting to the bump and the lower layer. **Note:** `The route_flip_chip` command considers the route width value from the constraint file if it is higher than `route_flip_chip -route_width`, to enable splitting of wires. *Data_type*: float, optional
`-selected_bumps`	Routes the selected bumps. The selected bumps can be either power bumps or signal bumps. *Data_type*: bool, optional
`-spread_wires_factor` *value*	Specifies automatic spreading of wires during bump routing in order to prevent any SI violations. The *value* is applied as a multiple of the minimum spacing that is required. *Data_type*: float, optional
`-straight_connections {straightwithdrcclean | straightwithchanges}`	
	Specifies that only straight connections are made between targets and that jogs are not allowed during routing. This option is available only with `route_flip_chip -target connect_bump_to_ring_stripe`. You must specify either the `-straight_connections` parameter or the `-jog_control` parameter. If you specify the `-straight_connections` parameter, you can also specify one or both of the following options: • `straightwithdrcclean`: Leaves a route open if a straight connection cannot reach a target without causing a DRC violation. If you do not specify this option, the software makes a straight connection to a target, even if the route creates a DRC violations. • `straightwithchanges`: Permits the route to change to another layer to avoid DRC violations. If you do not specify this option, the software makes only straight connections on the same layer. *Data_type*: enum, optional
`-target {connect_bump_to_ring_stripe | connect_bump_to_pad}`	
	Specifies the type of the bumps to be routed. **Note:** For `-target connect_bump_to_ring_stripe`: • If `-route_pg_style` is specified as `finger`, the power bumps are routed in the finger routing style. • If `-route_pg_style` is not specified or specified as `none`, the power bumps are routed in normal PG routing style. *Data_type*: enum, required
`-top_layer` *layerName*	Specifies the top-most metal layer number / `LEF` layer names that the software can use when routing bumps. *Default:* If you do not specify this parameter, all layers are used. *Data_type*: string, optional
`-user_class` *subclass_string*	
	Assigns specified string as a subclass name to the wires and vias created. *Data_type*: string, optional
`-verbose`	Specifies that each progress message appears in the log file and the Innovus console. *Data_type*: bool, optional

Related Information

- `check_flip_chip_constraints`
- `create_net_group`
- `route_point_to_point`
- `update_net_group`
- Attributes in the `flip_chip` category
- Flip Chip Methodologies chapter in the *User Guide*
- Tools Menu chapter in the *Menu Reference*
  - *Flip Chip*
  - *TSV*

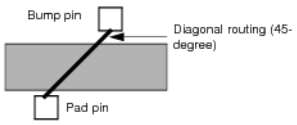
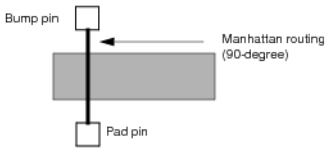
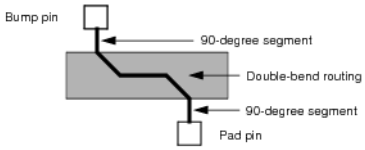
route_point_to_point

```
route_point_to_point
[-help]
[-constraint_file filename]
[-drop_via_in_trace]
[-nets {[netName | -1] [layerName | -1] [(x y)]}]
[-pins {[instName | -1] [pinName | -1] [(x y)]}]
[-route_layer {bot[:top[:step]] [,bot[:top[:step]]]}]
[-route_style {manhattan | double_bend | diagonal}]
[-spacing spacingValue]
[-split {layerBase maxWidth:value [gap:value] [style:RIVER|MESH]}]
[-user_class subclass_name]
[-width widthValue]
[{[-use_exact_location] [-guide_point {{x1 y1} {x2 y2}...} [-keep_partial]]} [-offset {x y}]]
```

Specifies routing constraints to perform point-to-point routing between I/O pad pins and bumps, and wires and bumps for SPECIALNETS (only) that are defined in the DEF file.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `route_point_to_point` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man route_point_to_point`
`-constraint_file fileName`	
	Specifies the name of the file that contains the nets defined in the DEF file. *Data_type*: string, optional
`-drop_via_in_trace`	Drops all possible vias in the trace. *Data_type*: bool, optional
`-guide_point {{x1 y1} {x2 y2}...}`	
	Specifies guiding points for routing. `x` and `y` values are real values in micron. You can specify up to 256 guiding points. The P2P router routes through all the usable guiding points in an order of distance to/from target. This feature allows you to control P2P router more precisely on pin selection. *Data_type*: (point)+, optional
`-keep_partial`	Keeps routed segments when routing with guide points fails partially. **Note:** This parameter can be used only with `-guide_point`. *Data_type*: bool, optional
`-nets {netName | -1 [layerName | -1] (x y)}`	
	Specifies net options, such as net name, net layer, and location coordinates. The `layerName` can now also be the TSV layer name. *Data_type*: string, optional
`-offset {x y}`	Defines the maximum search offset from the original user click location. **Note:** This parameter can be used only with `-use_exact_location`. *Data_type*: point, optional
`-pins {[instName | -1] [pinName | -1] [(x y)]}`	
	Specifies the pin options, such as the instance name, the pin name, or the location coordinates. *Data_type*: string, optional
`-route_layer {bot[:top[:step]] [,bot[:top[:step]]]}`	
	Specifies the route layer. *Data_type*: string, optional

`-route_style [manhattan | double_bend | diagonal]`	
	Specifies the routing style options for performing the point-to-point routing. • `diagonal`  • `manhattan`  • `double_bend`  *Data_type*: enum, optional
`-spacing spacingValue`	Specifies the route spacing, in micrometers. *Data_type*: float, optional
`-split {layerBase maxWidth:value [gap:value] [style:RIVER|MESH]}`	

	Splits the wire segment based on the maximum width and splitting style.
	The router gets the maximum width value from the layer information; the router does not split the wire segment when the wire width does not exceed the layer's maximum width.
	• `maxWidth:value`: `Specifies the maximum width for splitting.`
	• `gap:value`: Specifies the splitting gap.
	• `style:RIVER|MESH`:Specifies the splitting style:
	  ◦ `RIVER`: Allows only one crossing of the split wire.
	  ◦ `MASH`: Allows crossing of all of split wires and also drops all the crossing sections.
	Data_type: string, optional

`-use_exact_location`	Uses the exact location that you click to perform routing. You can use the optional -offset parameter to define the maximum search offset from the original user click location. If the location does not fulfill the pin access requirements, P2P router performs an auto search to find a new access point. **Note:** If the `-use_exact_location` parameter is not specified, P2P finds the start end locations on the user-clicked object on its own. *Data_type*: bool, optional
`-user_class subclass_name`	Assigns the specified string as a subclass name to the wires and vias created. *Data_type*: string, optional
`-width widthValue`	Specifies the route width, in micrometers. *Data_type*: float, optional

Examples

The following command performs diagonal style point-to-point routing between the I/O pad pin `newIo_2` and bump `Bump_41_0_4` on route layer `M8`, having a default route width of `0.44` micrometers, and a spacing of `0.46` micrometers:

```
route_point_to_point -route_layer M8:M8 -width 0.44 -spacing 0.46 -route_style diagonal -pins {newIo_2 -1 (-21.3115 535.932)} -pins
{Bump_41_0_4 port_pad_data_in[10] (76.507 482.032)}
```

Related Information

- `route_flip_chip`
- Flip Chip Methodologies chapter in the *User Guide*
  - P2P Router
- Tools Menu chapter in the *Menu Reference*
  - *TSV*

set_bond_pads_to_fixed

```
set_bond_pads_to_fixed
[-help]
{{-io_inst InstName [-pin pinName]} | -selected}
```

Assigns a fixed status to a specified bond pad. This restricts other bond pad commands from modifying the bond pad.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `set_bond_pads_to_fixed` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man set_bond_pads_to_fixed`
`-io_inst` *instName*	Specifies the name of the I/O instance whose bond pad will be assigned as fixed. *Data_type*: string, required
`-pin` *pinName*	Specifies the name of the pin inside the instance. *Data_type*: string, optional
`-selected`	Assigns a fixed status to a selected bond pad. You must first select the bond pad in the design display area, then use this parameter to assign the fixed status. *Data_type*: bool, required

Example

- The following command assigns a fixed status to the bond pad in I/O instance `pad_addr[30]`:
  ```
  set_bond_pads_to_fixed -io_inst {pad_addr[30]}
  ```

Related Information

- `check_bond_pad_spacing`
- `get_bond_pad`
- `place_bond_pad`
- `place_bond_pad_by_stagger`
- `reset_fixed_bond_pad`
- `space_bond_pads`

set_bump_master

```
set_bump_master
[-help]
-cell bumpMasterName
[-from_bump_cell bumpMasterName]
[-location_type {cell_center cell_lowerleft geometry_center geometry_lowerleft}]
{-net netName | -all | -bumps bump_list | -selected}
```

Enables you to replace the cell master for specified bumps.

Parameters

`-help`	Prints a brief description that includes type and default information for each `set_bump_master` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man set_bump_master`
`-all`	Specifies that all bumps will use the new cell master specified using the `-cell` parameter. *Data_type*: bool, optional
`-bumps bump_list`	Specifies the list of bumps for which you want to change the bump master. Use this parameter if you want to change the cell master of specific bumps, which may or may not have the same cell master originally. *Data_type*: string list, optional
`-cell bumpMasterName`	
	Specifies the name of the new cell master. *Data_type*: string, required
`-from_bump_cell bumpMasterName`	
	Specifies the name of the old cell master. Use this parameter to replace the cell master of bumps using a specific bump master. *Data_type*: string, optional
`-location_type {cell_center cell_lowerleft geometry_center geometry_lowerleft}`	
	Specifies the location type. *Default*: `cell_lowerleft` *Data_type*: enum, optional
`-net netName`	Specifies that all bumps connected to the specified net will use the new cell master. For example, if the *netName* is VDD, all connected bumps will use the cell master specified using the `-cell` parameter. *Data_type*: string, optional
`-selected`	Changes the bump master of selected bumps to the new cell master as specified using the `-cell` parameter. *Data_type*: bool, optional

Examples

- The following example changes the cell master of selected bumps from `BUMPCELL` to `BGA_PAD`:
  ```
  set_bump_master -cell BGA_PAD -selected -from_bump_cell BUMPCELL
  ```
  Note: The change happens only for those of the selected bumps that originally belonged to `BUMPCELL`. If a selected bump belongs to a cell master other than `BUMPCELL`, its cell master is not changed.
- The following command changes the cell master of all the bumps connected to the net `int` to `BUMP_PWR`:
  ```
  set_bump_master -cell BUMP_PWR -net int
  ```
- The following command changes the cell master of all the bumps in the design to `BUMP_PWR`:
  ```
  set_bump_master -cell BUMP_PWR -all
  ```
- The following command changes the the cell master of all the bumps connected to the net `int` to `BUMP_PWR` and specifies their location type as `geometry_center`:
  ```
  set_bump_master -cell BUMP_PWR -net int -location_type geometry_center
  ```

Related Information

- Flip Chip Methodologies chapter in the *User Guide*

set_bumps_fixed_assign

```
set_bumps_fixed_assign
[-help]
{-all | -bumps {list} | -cells {list}}
```

Fixes a specified group of assigned bumps to keep them from being reassigned. The software updates the DEF file by adding the `+FIXEDBUMP` attribute to the specified group of assigned bumps to indicate that they are now in a fixed state. You can use the `write_def` command to view the bumps with the `+FIXEDBUMP` attribute. Use this command after bumps are assigned with the `assign_bumps` command or defined in the I/O file with the signal name.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `set_bumps_fixed_assign` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man set_bumps_fixed_assign`.
`-all`	Fixes all of the assigned bumps. *Data_type*: bool, optional
`-bumps list`	Fixes the assigned bumps that are specified in the bump name *list*. *Data_type*: string, optional
`-cells list`	Fixes the assigned bumps that are specified in the bump site *list*. *Data_type*: string, optional

Example

- The following command fixes all of the assigned bumps:
  ```
  set_bumps_fixed_assign -all
  ```

Related Information

- `assign_bumps`
- `set_bumps_place_status`
- `reset_bumps_fixed_assign`
- `write_def`
- Flip Chip Methodologies chapter in the *User Guide*
- Tools Menu chapter in the *Menu Reference*
  - *Flip Chip*

set_bumps_place_status

```
set_bumps_place_status
[-help]
{-bump Name | -selected}
-status {placed fixed cover}
```

Sets the bump placement status.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `set_bumps_place_status` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man set_bumps_place_status`.
`-bump Name`	Specifies the name of the bump to set the placement status. *Data_type*: string, required
`-selected`	Specifies to set the bump placement status for the selected bump. *Data_type*: bool, required
`-status {placed fixed cover}`	Specifies the bump placement status. *Data_type*: enum, required

Example

- The following command sets the placement status of the bump `Bump_53_2_5` to `fixed`:
  ```
  set_bumps_place_status -status fixed -bump Bump_53_2_5
  ```
  Note: You can also double-click the bump to change the placement status from `placed` to `fixed`.

Related Information

- `set_bumps_fixed_assign`
- `reset_bumps_fixed_assign`
- Flip Chip Methodologies chapter in the *User Guide*
- Data Preparation chapter in the *User Guide*
  - *Defining BUMP CELL Placement Status*
- Tools Menu chapter in the *Menu Reference*
  - *Flip Chip*

set_probe_pin

```
set_probe_pin
[-help]
{{-inst instName -pin pinName} | -bump bumpName}
```

Sets a pad or bump as a probe point. Probe points are used for package-level design and testing. To delete or report on probe points, use the `reset_probe_pin` or `report_probe_pins` commands, respectively.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `set_probe_pin` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man set_probe_pin`
`-bump bumpName`	Specifies the instance name of the bump. *Data_type*: string, required
`-inst InstName`	Specifies the name of the I/O instance. *Data_type*: string, required
`-pin pinName`	Specifies the name of the pin on the I/O instance that will be set as a probe point. *Data_type*: string, required

Example

- The following command sets pin `PAD` of I/O instance `Bump_1` as a probe point:
  ```
  set_probe_pin -inst Bump_1 -pin PAD
  ```

Related Information

- `report_probe_pins`
- `reset_probe_pin`

set_special_route_type

```
set_special_route_type
[-help]
[-stop_layer Metal_layer]
-stop_type_list swire_type_list
-type {blockring blockagewire blockwire corewire drcfill fillwireopc fillwire followpin iowire notype padring ring stripe}
{-bump_cell bumpCellList | -pin {{inst1 pin1} {inst2 pin2}...}}
```

Sets special routing wires to a particular shape type so that they can be written out into a DEF file using the `-special_shape` parameter of the `write_def_by_section` command. This makes data filtering easier by enabling customized selection and output of special routing shape types. The output data is very specific and can be easily managed as storage size is small.

The `set_special_route_type` command works by tracing special wires and vias that touch bumps or instance pins through to a stop-layer or special wire stop type. All the traced special wires and vias are marked with a particular wire type. The `write_def_by_section -special_shape` command is then used to write out the special type routes and vias.

Note: Use the `set_special_route_type` and `write_def_by_section` commands in sequence. Save the design before you modify the shape types so that you can recover the original design with `read_def`.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `set_special_route_type` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man set_special_route_type`
`-bump_cell` *`bumpCellList`*	Specifies the list of bump cells. If this option specified, the tracing starts from any special wires and vias that touch the named bump cells. **Note:** This option cannot be used with `-pin`. *Data_type*: string list, required
`-pin {{`*`inst1 pin1`*`} {`*`inst2 pin2`*`}...}`	
	Specifies a list of pairs of instances and their pins. If this option specified, the tracing starts from any special wires and vias that touch the named instance pin pairs. **Note:** This option cannot be used with `-bump_cell`. *Data_type*: string list, required
`-stop_layer` *`Metal_layer`*	Specifies the metal layer at which tracing should stop. *Data_type*: string, optional
`-stop_type_list` *`swire_type_list`*	
	Specifies the special-wire types used in tracing. When a special wire with a shape type that is not included in the list is encountered, tracing stops. The legal list of *`swire_type_list`* are the DEF SPECIAL ROUTE SHAPE types. *Data_type*: string list, required
`-type {blockring blockagewire blockwire corewire drcfill fillwireopc fillwire followpin iowire notype padring ring stripe}`	
	Sets the marked special wire to the specified wire type (DEF SPECIAL ROUTE SHAPE type). *Default*: `iowire` *Data_type*: enum, required

Example

- The following command traces special wires and vias that touch the named bump cells. Tracing stops when a special wire of type other than `iowire` is encountered. The marked special wire is set to `iowire`.

```
set_special_route_type -bump_cell {bc*} -stop_type_list {iowire} -type iowire
```

Related Information

- `read_def`
- `write_def_by_section`

space_bond_pads

```
space_bond_pads
[-help]
[-spread]
```

Spaces bond pads and the corresponding area I/O instances based on the constraints returned by the `getCellDist()` Tcl script.

The command distributes excess space until all the bond pads reach their maximum spacing constraint returned by the `getCellDist()` Tcl script or until the time when no extra space is left between the instances.

The initial distribution of space occurs for the clusters of instances that have the first or the last bond pads in the row. The remaining iterations equally distribute the remaining space between each I/O. The I/O instances that are in the end are placed at minimum spacing to the die edges, both on the left and right sides of the die.

The I/O pads will snap to I/O sites and the corresponding bond pads (bumps) will shift accordingly.

You can define the constraints in the `getCellDist()` Tcl script as follows:

```
proc getCellDist { xp yp xn yn padPosition padCount side ioRow staggerPos ioName }
```

Here:

- *xp yp* - Specifies the x and y coordinates of a previous bond pad having the same stagger position. If first, the location is 0, 0.
- *xn yn* - Specifies the x and y coordinates of the previous neighboring bond pad having any stagger position. If first, the location is 0, 0.
- *padPosition* - Specifies the position of the bond pad. The position can be i (inner), m (middle), or o (outer).
- *padCount* - Specifies the current bond pad count for bond pads at the same stagger position. For example, 1 for the first inner bond pad, 2 for the second inner bond pad, and so on.
- *side* - Specifies the n (North), s (South), e (East), or w (West) side of the bond pad.
- *ioRow* - Specifies the I/O row.
- *staggerPos* - Specifies the stagger type.
- *ioName* - Specifies the name of the I/O.

Note:

- The bond pad spacing for North and South is done from left to right.
- The bond pad spacing for East and West is done from bottom to top.
- The bond pad spacing for each side and ring is done independently.

After using the `spaceBondPad` command, you can use the `check_bond_pad_spacing` command to check the bond pad spacing.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `space_bond_pads` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man space_bond_pads`
`-spread`	Removes any extra spacing that may exist between I/O cells, which were placed based on the rules defined in the `getCellDist()` Tcl script, and evenly distributes the extra space between all of the I/O cells in that row. You can use this parameter if there are large gaps (spaces) between I/O cells. *Data_type*: bool, optional

Related Information

- check_bond_pad_spacing
- get_bond_pad
- place_bond_pad
- place_bond_pad_by_stagger
- reset_fixed_bond_pad
- set_bond_pads_to_fixed

swap_bump_assignment

```
swap_bump_assignment
[-help]
[[-from {nameList} -to {nameList}] | -circle_offset value]
```

Swaps signal between selected instances (I/O pads or bumps).

To swap bump assignments in a circle, determine the value of offset you want to use. Assuming that the value of offset is `N`, the number of the selected bumps is `M`, the order of the input bump is `$index`, and the name of the bump with `$index` is `bump_$index`, you can swap bump assignments in a circle by using the the following command:

```
swap_bump_assignment -from {bump_1 bump_2 bump_3 ... bump_$index} -to {bump_[1+N] bump_[2+N] bump_[3+N]... bump_[$index+N-M]}
```

Parameters

`-help`	Prints a brief description of the `swap_bump_assignment` command. For a detailed description of the command, use the `man` command: `man swap_bump_assignment`
`-circle_offset value`	Specifies the offset value to swap bumps in circle. The value should be be positive and less than the number of the selected bumps. For example, if you select the bumps `bump1`, `bump2`, `bump3`, `bump4`, and `bump5` in the GUI and specify `-circle_offset` as `2`, the result of signal swapping will be as follows: `bump1(S1) bump2(S2) bump3(S3) bump4(S4) bump5 (S5) -> bump1(S4) bump2(S5) bump3(S1) bump4(S2) bump5 (S3)` *Data_type*: int, optional
`-from nameList`	Specifies the names of bumps from which signal is to be swapped. *Data_type*: string, optional
`-to nameList`	Specifies the names of bumps to which signal is to be swapped. *Data_type*: string, optional

Related Information

- `assign_bumps`
- `unassign_bumps`
- `gui_show_bump_connections`
- Flip Chip Methodologies chapter in the *User Guide*
- Tools Menu chapter in the *Menu Reference*
  - *Flip Chip*

unassign_bumps

```
unassign_bumps
[-help]
{-all | -bumps {list} | -cells {list} | -selected}
```

Removes the Verilog signal names from bumps, which removes the connection between bumps and I/O pins. The original connection will remain in the Verilog and is connected when the bump is reassigned. If the bump is assigned, it will appear in the DEF `SPECIALNETS` section with the signal (net) name. Use this command after bumps are assigned with the `assign_bumps` command or defined in the I/O file with the signal name, but without the `-fixed` constraint.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `unassign_bumps` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man unassign_bumps`
`-all`	Removes all the connections between bumps and I/O pins and removes the signal assignment of the bumps. *Data_type*: bool, optional
`-bumps list`	Removes only the connections between bumps and I/O pins contained in the bump name list. *Data_type*: string, optional
`-cells list`	Removes only the connections between bumps and I/O pins contained in the site list. *Data_type*: string, optional
`-selected`	Specifies to unassign the bump(s) selected in the design in the GUI. *Data_type*: bool, optional

Examples

- The following command unassigns all of the bumps:
  ```
  unassign_bumps -all
  ```
- The following command unassigns the (named) bumps `bumpA` and `bumpB`:
  ```
  unassign_bumps -bumps {bumpA bumpB}
  ```

Related Information

- `assign_bumps`
- `check_bumps`
- `delete_bumps`
- Flip Chip Methodologies in the *User Guide*
- Tools Menu chapter of the *Menu Reference*
  - *Flip Chip*

update_bump_connect_target_constraints

```
update_bump_connect_target_constraints
[-help]
{-bumps list_of_bumps | -selected }
{-io_inst instance_name
  {[-to_io_inst instance_name]
   [-pin_name pin_name {[-to_pin_name pin_name]
                       [{-port_num value -to_port_num value}]}]}}
```

Modifies the property values on bumps. You can change the instance name, pin name, and port name on specified or selected bumps.

Parameters

`-help`	Prints a brief description that includes type and default information for each `update_bump_connect_target_constraints` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man update_bump_connect_target_constraints`
`-bumps list_of_bumps`	Edits the bump connect target constraint on specified bumps. The `-bumps` option accepts wild cards. *Data_type*: string list, required
`-io_inst instance_name`	Specifies the name of the instance for which you want to change the bump connect target constraint. *Data_type*: string, required
`-pin_name pin_name`	Specifies the name of the pin for which you want to change the bump connect target constraint. *Data_type*: string, optional
`-port_num value`	Specifies the port number for which want to change the bump connect target constraint. *Data_type*: int, optional
`-selected`	Edits the bump connect target constraint on selected bumps. Multiple bumps can be selected. *Data_type*: bool, required
`-to_io_inst instance_name`	Specifies the instance name to change to in the bump connect target constraint. *Data_type*: string, optional
`-to_pin_name pin_name`	Specifies the pin name to change to in the bump connect target constraint. *Data_type*: string, optional
`-to_port_num value`	Specifies the port number to change to in bump connect target constraint. *Data_type*: int, optional

Example

- The following command modifies `bump_connect_target` property on `Bump_1` to change pairing between `Bump_1` and instance `IOPADS_INST/esd` from `PORT1` to `PORT2`. `PORT1` and `PORT2` are under `VDD` PIN of instance `IOPADS_INST/esd`.
  ```
  update_bump_connect_target_constraints -bumps Bump_1 -io_inst IOPADS_INST/esd -pin_name VDD -port_num 1 -to_port_num 2
  ```

Related Information

- `create_bump_connect_target_constraint`
- `delete_bump_connect_target_constraints`
- `get_pin_port_number`
- Flip Chip Methodologies chapter of the *User Guide*
  - *Port Numbering*

- Tools Menu chapter in the *Menu Reference*
  - *Flip Chip*

update_bump_name

```
update_bump_name
[-help]
-from current_pattern
[-selected]
-to new_pattern
```

Modifies the names of selected or specified bumps according to the specified pattern.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `update_bump_name` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man update_bump_name`
`-from current_pattern`	
	Specifies the name of a single bump or a pattern for the names of the bumps to be edited. The pattern is a regular expression, such as `{Bump_\d+}` or `{Bump_(\d)(\w+)}`, which can be referred in TCL. When specifying a pattern, the opening and closing braces are required. *Data_type*: string, required
`-selected`	Modifies the names of selected bumps. *Data_type*: bool, optional
`-to new_pattern`	Specifies the new name or name format for target bumps. When specifying a name format, the opening and closing braces are required. *Data_type*: string, required

Examples

- The following command modifies the names of all bumps that have the pattern `Bump_\d+`, such as `Bump_1` and `Bump_12`, to bumps with pattern `signal_\d+`, such as `signal_1` and `signal_12`:
  ```
  update_bump_name -from {Bump_} -to {signal_}
  ```
- The following command modifies the name of the `Bump_1_1_1` bump to `signal_bump`:
  ```
  update_bump_name -from {Bump_1_1_1} -to {signal_bump}
  ```
- The following command modifies the names of all bumps that have the pattern `Bump_(\d)_(\d)_(\d)` to bumps named `1_1_1_signal`, `1_1_2_signal`, and so on:
  ```
  update_bump_name -from {Bump_(\d)_(\d)_(\d)} -to {\1_\2_\3_signal}
  ```

Related Information

- `copy_bumps`
- `create_bump`
- `delete_bumps`
- `set_bump_master`

write_flip_chip_property

```
write_flip_chip_property
[-help]
out_file
```

Saves the bump and pad properties, including virtual connection area information, in the bump and IO pad property file. The property file is a manually editable text file. Virtual connection area records are separated by new lines in the file.

Parameters

`-help`	Prints out the command usage that includes type and default information for each `write_flip_chip_property` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man write_flip_chip_property`
`out_file`	Specifies the bump and pad property file name. *Data_type*: string, required

Example

The following command saves the bump and pad properties, including virtual connection area information, in a file named `test2`:

```
write_flip_chip_property test2
```

Related Information

- `read_io_update`
- `read_flip_chip_property`
- Flip Chip Methodologies chapter of the User Guide
- Tools Menu chapter in the Menu Reference
  - *Flip Chip*

11

Floorplan Commands and Attributes

- add_area_io_fillers
- add_area_io_rows
- add_dummy_boundary_wires
- add_fences
- add_interposer_route_block
- add_io_fillers
- add_io_insts
- add_io_row_fillers
- add_physical_cell
- align_selected
- check_floorplan
- check_floorplan_space
- check_macro_lower_left_on_track
- check_macro_place_constraint
- check_ufc
- convert_fence_to_lef
- copy_obj
- create_boundary_constraint
- create_bump_grid
- create_exclusive_groups
- create_floorplan
- create_group
- create_interposer_route_block
- create_io_row
- create_logic_hierarchy
- create_pg_model_for_macro_place
- create_pg_pin
- create_physical_pin
- create_place_blockage
- create_place_halo
- create_relative_floorplan
- create_resize_blockage
- create_route_blockage
- create_route_blockage_along_track
- create_route_halo
- create_route_rule
- create_row
- create_shape

- create_special_drc_region
- create_special_mark_layer_shape
- create_text
- delete_all_density_area
- delete_all_floorplan_objs
- delete_all_power_preroutes
- delete_all_signal_preroutes
- delete_bump_grid
- delete_exclusive_groups
- delete_io_fillers
- delete_io_insts
- delete_io_row_fillers
- delete_pg_pins
- delete_place_halo
- delete_relative_floorplan
- delete_route_blockages
- delete_route_halos
- delete_row
- delete_selected_from_floorplan
- finish_floorplan
- finish_floorplan Category Attributes
- fix_boundary_overlaps
- fix_floorplan
- flip_or_rotate_obj
- floorplan Category Attributes
- get_draw_view
- get_io_flow_flag
- get_macro_place_constraint
- gui_set_draw_view
- gui_show_hinst_in_floorplan
- gui_show_obj_connectivity
- init_core_rows
- legalize_floorplan
- move_group_pins
- move_macro_in_hinsts
- move_obj
- paste_obj
- place_area_ios
- place_blocks_refine
- place_macro_detail
- place_pad_ios
- place_peripheral_ios
- plan_design Category Attributes

- read_floorplan
- read_io_file
- report_floorplan_obj
- report_narrow_channel
- report_net_groups
- report_obj_connectivity
- report_off_grid_blocks
- report_seed_connections
- reset_fixed_module_size
- reset_macro_place_constraint
- resize_floorplan
- resize_floorplan_channel
- resize_floorplan Category Attributes
- resize_partition_to_abut
- set_all_ios_to_fixed
- set_bottom_io_pad_orient
- set_flipping
- set_floorplan_row_spacing_and_type
- set_inst_group_physical_hier
- set_instance_placement_status
- set_io_constraints
- set_io_flow_flag
- set_io_row_margin
- set_macro_place_constraint
- set_module_size_fixed
- set_resize_line
- set_route_blockage_default_layer
- snap_floorplan
- snap_floorplan_io
- space_selected
- split_rectilinear_inst
- split_row
- stretch_rows
- swap_pins
- trace_obj_connectivity
- trace_obj Category Attributes
- unplace_obj
- update_floorplan
- update_floorplan_for_macro_place
- update_floorplan_obj
- update_group
- update_obj_floorplan_cut_box
- update_origin

- update_partition_to_stairway
- update_route_rule
- write_floorplan
- write_floorplan_script
- write_io_file
- write_macro_place_constraint
- write_route_rule

add_area_io_fillers

```
add_area_io_fillers
[-help]
{-cell fillerCellName | -cells {fillerCellName1 fillerCellName2 ...}}
[-prefix prefix]
[-area_io_row_cluster aioRowClusterName | -all_area_io_row_cluster]
[-set_preplaced]
[-only_esd]
[-only_shoulder]
[-only_gap]
[-power_domain powerDomainName]
```

Adds area I/O filler cell instances. Use this command during or after floorplanning. You can also use the command after the placement of manually created area I/Os.

The command,

- by default, adds area I/O filler cells to all the area I/O clusters if you do not specify the `-area_io_row_cluster` and `-all_area_io_row_cluster` parameters.
- adds filler cells in all the blank sites of the specified area I/O row clusters if you do not specify the `-only_esd`, `-only_gap` and `-only_shoulder` parameters.
- uses any LEF `CLASS PAD` or `CLASS PAD AREAIO` for adding the area I/O filler cells. By default the command uses LEF `CLASS PAD SPACER` to add filler cells, and also adds filler cells using any LEF `CLASS PAD` displaying a warning in Innovus.
- does not allow overlapping of filler cells and places non-overlapping filler cells.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `add_area_io_fillers` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man add_area_io_fillers`
`-all_area_io_row_cluster`	Adds AIOFiller cells to all the area I/O clusters. *Data_type*: bool, optional
`-area_io_row_cluster` *aioRowClusterName*	Specifies the area I/O row cluster name. AIO filler cells are added only to this cluster. *Data_type*: string, optional
`-cell` *fillerCellName*	Specifies the name of the area I/O filler cell to be added. *Data_type*: string, optional
`-cells {`*fillerCellName1 fillerCellName2 ...*`}`	Specifies the list of area I/O filler cells that fill the gaps in I/O row clusters. Adds the largest filler cell first, followed by smaller filler cells. *Data_type*: string, optional
`-only_esd`	Specifies to add ESD (electrostatic discharge) cells with the filler cell name in the area I/O row clusters. *Data_type*: bool, optional
`-only_gap`	Specifies to add the filler cells with the filler cell name to fill all the gaps in the area I/O row clusters. *Data_type*: bool, optional
`-only_shoulder`	Specifies to add shoulder cells with the filler cell name in the area I/O row clusters. *Data_type*: bool, optional
`-power_domain` *powerDomainName*	Specifies the name of the power domain in which the area I/O filler cells are to be added. *Data_type*: string, optional
`-prefix` *prefix*	Specifies the prefix name to be appended for the added area I/O filler cell instance. *Data_type*: string, optional
`-set_preplaced`	Sets the status of the area I/O filler cells as preplaced. *Data_type*: bool, optional

Examples

- The following command fills all area I/O row clusters on `ESD`, `shoulder`, and `gap` sites with a filler cell `A0_123` with preplaced status:
  ```
  add_area_io_fillers -cell A0_123 -set_preplaced
  ```
- The following command adds area I/O filler cells,`IOFILL` and `IOFILL_HALF`,to fill all the gaps in all area I/O row clusters:
  ```
  add_area_io_fillers -all_area_io_row_cluster -only_gap -cells { IOFILL IOFILL_HALF }
  ```

Related Information

- `add_area_io_rows`
- `add_io_fillers`
- `add_io_insts`
- `add_io_row_fillers`
- `create_io_row`
- Floorplanning the Design chapter in the *Innovus Stylus Common UI User Guide*

add_area_io_rows

```
add_area_io_rows
[-help]
-site site_name
{-bump bumpId | -location x y}
[-dx x_dist] [-dy y_dist]
[-orient {r0|r90|r180|r270|mx|mx90|my|my90}]
[-horizontal | -vertical]
[-esd esites_no...]
[-shoulder site_no...]
[-cluster_halo space]
[-num num]
[-abut_row]
[-row name]
[-no_snap]
[-soft]
[{-row_cluster_array_x elem1 | -row_cluster_array_y elem2} {-pitch_x dist | -pitch_y dist}]
```

Adds area I/O rows. Use this command during or after floorplanning.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `add_area_io_rows` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man add_area_io_rows`
`-abut_row`	When creating two I/O rows that abut each other, this option specifies that one of them will be in a flip orientation to the other. *Data_type*: bool, optional
`-bump bumpId`	Specifies the bump ID that is used to specify the start point of the row area. The `-bump` parameter creates IO rows from the center point of the specified bump. For example, when you specify -bump 17_1_1, the row will be created from the center point of this bump. The length of row is controlled by other options. *Data_type*: string, required
`-cluster_halo space`	Specifies the distance, in microns, around the boundary of a row cluster to prevent other cells from being placed near it. *Data_type*: float, optional
`-dx x_dist`	Offsets the reference point, in microns, to the first I/O row on the `x` axis. *Data_type*: float, optional
`-dy y_dist`	Offsets the reference point, in microns, to the first I/O row on the `y` axis. *Data_type*: float, optional
`-esd esites_no...`	Reserves the sites in the row as electrostatic discharge cells (ESD). For example: `-esd 2,7` **Note:** Use only when you create cluster rows with multiple sites. **Note:** Do not use the `-soft` option with this option. *Data_type*: string, optional
`-horizontal`	Creates horizontal rows. *Data_type*: bool, optional
`-location x y`	Specifies the lower left `x` and `y` coordinate reference points, in microns, of the new I/O row. *Data_type*: point, required

`-no_snap`	Adds the area IO row at the exact location specified by the `add_area_io_rows -location` coordinates rather than snapping the row to the standard cell placement grid. *Data_type*: bool, optional
`-num` *num*	Specifies the number of sites. *Data_type*: int, optional
`-orient {r0|r90|r180|r270|mx|mx90|my|my90}`	Specifies the orientation for the row. See *Orientation Key* below for more information. *Default*: `r0` *Data_type*: enum, optional
`-pitch_x` *dist*	Specifies the distance between bump centers, in microns, in the `x` direction. *Data_type*: float, optional
`-pitch_y` *dist*	Specifies the distance between bump centers, in microns, in the `y` direction. *Data_type*: float, optional
`-row` *name*	Specifies a user-defined instance name for an added IO Row or a prefix for multiple rows, as in a matrix. If only one row is added, the name is used exactly as specified. If multiple rows are added, a number prefix is appended to the name. *Data_type*: string, optional
`-row_cluster_array_x` *elem1*	Specifies the x dimensions of the row cluster array size. For example, to create a 5 x 8 row array, `-row_cluster_array_x 5 -row_cluster_array_y 8` *Data_type*: int, optional
`-row_cluster_array_y` *elem2*	Specifies the y dimensions of the row cluster array size. For example, to create a 4 x 6 row array, `-row_cluster_array_x 4 -row_cluster_array_y 6` *Data_type*: int, optional
`-shoulder` *site_no...*	Reserves the sites in a row as shoulder cells. For example: `-shoulder 1,10` **Note:** Use only when you create cluster rows with multiple sites. **Note:** Do not use `-soft` option with this option. *Data_type*: string, optional
`-site` *siteName*	Specifies the site name for the area I/O row. *Data_type*: string, required
`-soft`	Creates a reef or cluster that can be collapsed if no I/O cells are placed in the available spaces within the reef or cluster. Use this parameter if the cluster site is not fully placed so that standard cells can be placed in these spaces if I/O cells are not placed there. **Note:** Do not use the `-esd` or `-shoulder` options with this option. *Data_type*: bool, optional
`-vertical`	Creates vertical rows. *Data_type*: bool, optional

Orientation Key

The following table is a key to row orientation:

Value	Definition

`r0`	No rotation
`mx`	Mirror through X axis
`my`	Mirror through Y axis
`r180`	Rotate counter-clockwise 180 degrees
`mx90`	Mirror through X axis and rotate counter-clockwise 90 degrees
`r90`	Rotate counter-clockwise 90 degrees
`r270`	Rotate counter-clockwise 270 degrees
`my90`	Mirror through Y axis and rotate counter-clockwise 90 degrees

Examples

- The following command creates an I/O row with site name `IO1` at the location related to the center of the bump selected in the design window. There is a 5 x 7 row array and each element in the array is separated by `100` µm in both the `x` and `y` axes. Each row has `8` sites.

  ```
  add_area_io_rows -site IO1 -bump 0x33457a -horizontal -num 8 -row_cluster_array_x 5 -row_cluster_array_y 7 -pitch_x 100 -pitch_y 100
  ```

- The following command creates a soft I/O row cluster with a `15.0` micron halo around it:

  ```
  add_area_io_rows -site IO3 -location 0.0 0.0 -dx 230 -dy 58.8 -num 7 -orient r0 -vertical -soft -abut_row -cluster_halo 15.0
  ```

- The following command creates an IO Row with a user-defined name:

  ```
  add_area_io_rows -site IO1 -row newrow -location 0 0 -dx 0 -orient r0 -vertical -num 1
  ```

Related Information

- `add_area_io_fillers`
- `add_io_fillers`
- `add_io_insts`
- `add_io_row_fillers`
- Floorplanning the Design chapter in the *Innovus Stylus Common UI User Guide*

add_dummy_boundary_wires

```
add_dummy_boundary_wires
[-help]
-layers layer_name | { layer_name_list ... }
[-masks mask_number | { mask_number_list ... }]
[-shape {blockring | blockagewire | blockwire | corewire | drcfill| fillwireopc | fillwire | followpin | iowire | none | padring | ring |
stripe} ]
[-short_wires wire_length | { wire_length_list ... }]
[-space space_value | { space_value_list ... }]
```

Add wires on the first and last tracks in the direction parallel to preferred routing direction with specified layers.

Note: DRC rules define forbidden regions near the design boundary edges that are parallel to the preferred routing direction of a layer. Any wires in these regions cannot project to the design boundary directly. Using the `add_dummy_boundary_wires` command you can add wires that shadow the design boundary such that no routing wires in the forbidden regions can project to the design boundary. This helps to avoid DRC violations.

Note: The `add_dummy_boundary_wires` command enables long dummy wires to honor route blockages by default.

Note: The `add_dummy_boundary_wires` command honors routing blockages and adds dummy wires along the routing blockages when the track box overlaps with them. The dummy wires do not cross partial routing blockages around the design boundary. It keeps the wires at the boundary and discontinues them when there is a routing blockage.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `add_dummy_boundary_wires` parameter. For a detailed description of the command and all of its parameters, use the man command: `man add_dummy_boundary_wires`
`-layers layer_name | { layer_name_list ... }`	

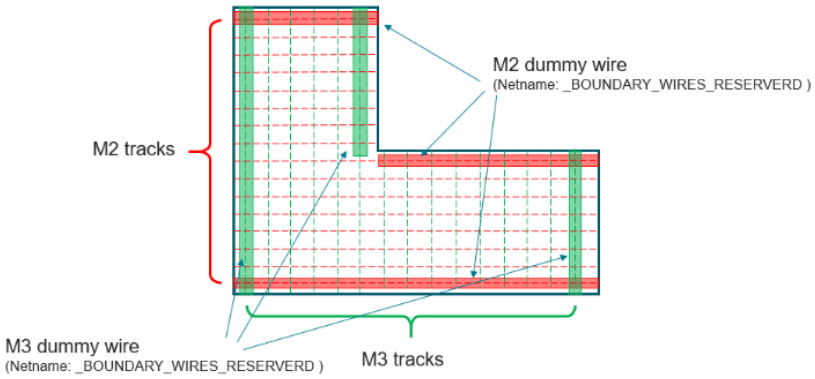
	Specifies the layers on which the dummy boundary wires are added.
	Note: The net name of the dummy wires is "`_BOUNDARY_WIRES_RESERVED`".
	Note: You can specify the *layerName* as a number, abbreviation or as a layer name defined in LEF. For example, *layerName* can be specified as 2, M2 or Metal2.
	`add_dummy_boundary_wires -layers {metal2 metal3} //full name works`
	`add_dummy_boundary_wires -layers {M2 M3} //abbreviated name works`
	`add_dummy_boundary_wires -layers {2 3} ///layer number works`
	Data_type: string, required

`-masks mask_number | { mask_number_list ... }`	Specifies one or multiple mask numbers which short dummy boundary wires snap to. The number of specified mask values should match the number of the specified layers in option layers. *Data_type*: string, optional
`-shape {blockring | blockagewire | blockwire | corewire | drcfill| fillwireopc | fillwire | followpin | iowire | none | padring | ring | stripe}`	
	Specifies the shape of the dummy boundary wires. *Default:* `none` *Data_type*: string, optional
`-short_wires wire_length | { wire_length_list ... }`	Specifies one or multiple set of lengths of short dummy boundary wires and space between them. The number of specified shortWires values should be twice of the number of specified layers in option layers. *Data_type*: string, optional
`-space space_value | { space_value_list ... }`	Specifies one or multiple space values to design boundary. **Note**: The space value is the distance between the design boundary and the start or end of the wire for 90 corner. The space value does not apply to 270 corner. *Default:* `0` *Data_type*: string, optional

Examples

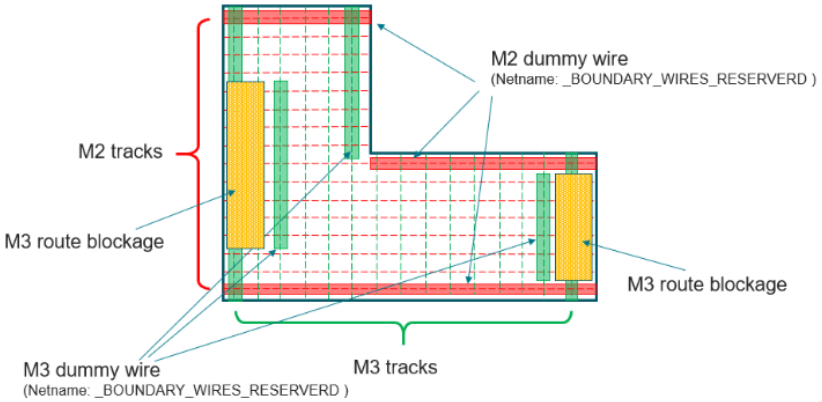
- The following command adds dummy wires on layers metal2 and metal3:

```
add_dummy_boundary_wires -layers {M2 M3}
```



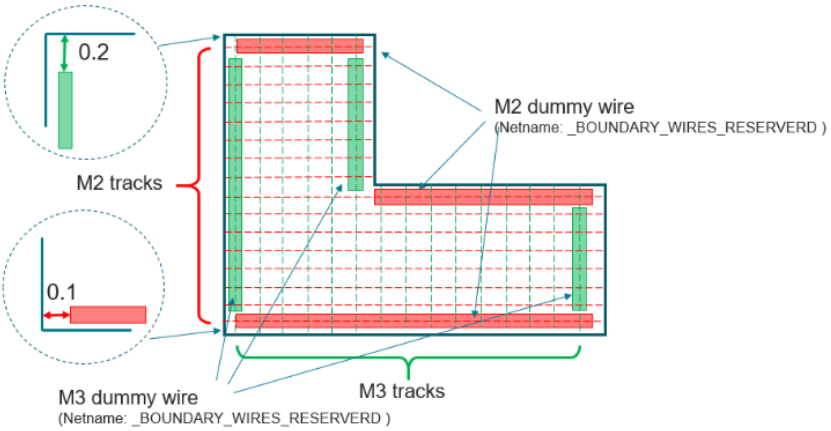
- The following command adds dummy wires on layers metal2 and metal3. Dummy wires will honor route blockages by default on Metal3 :

```
add_dummy_boundary_wires -layers {M2 M3}
```



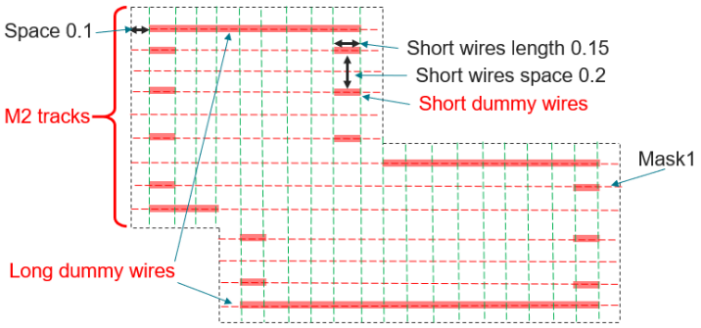
- The following command adds dummy wires with space value on layers metal2 and metal3:

```
add_dummy_boundary_wires -layers {M2 M3} -space {0.1 0.2}
```



- The following command adds long dummy wires with specified space and short dummy wires with specified length, space and mask on layers metal2:

```
add_dummy_boundary_wires -layers M2 -space 0.1 -short_wires {0.15 0.2} -masks 1
```



Related Information

- `legalize_floorplan`
- Floorplanning the Design chapter in the *Innovus Stylus Common UI User Guide*

add_fences

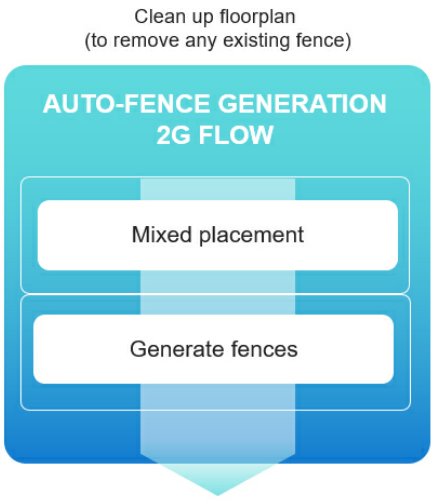
```
add_fences
[-help]
[-flow {2G}]
[-for_partition]
[-rectangle_shape]
[-target_density float]
[ [-hinst {hinst(s)}] | [-module {module(s)}] | [-inst_group {inst_group(s)}] ]
```

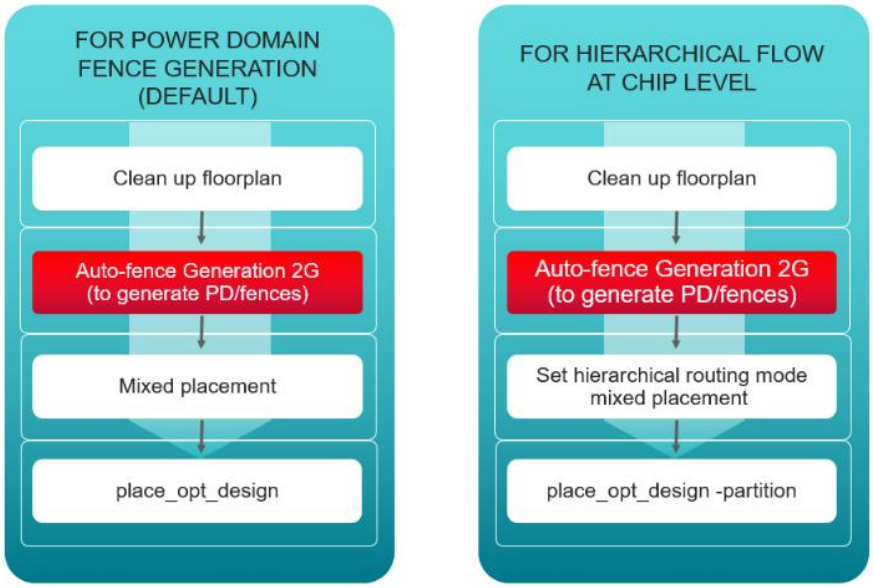
Automatically draws partition fences that enclose all their children flexModel guides. It helps to improve the usability of the prototyping flow. The recommended use model for automatically generating fences is as follows:

1. Run `proto_design` to place flexModel guides/fences.
2. Manually adjust/move flexModels to shorten timing paths.
3. Run the following commands to analyze the floorplan and check timing. Repeat step 2 and 3 if needed
   a. `place_opt_design`
   b. `route_early_global`
   c. `time_design -proto`
   d. `read_timing_debug_report -proto`
4. Select flexModels for partitions and define partitions.
5. Run `add_fences` to create fences that enclose all their children flexModel guides.

Note: If no option is specified then the command draws fences for all existing defined partitions.

Parameters

`-flow {2G}`	Enables the Auto-Fence Generation 2G flow to draw the fence shapes automatically. When this flow is enabled, the mixed-placement capability is leveraged to give optimal module placement distribution. A fence is generated for each target module after considering target utilization, boundary gaps and I/Os. The Auto-Fence Generation 2G flow involves the following key steps:  • Running flat global mixed placement • Creating the fence based on the module cells placement distribution and control the utilization. • Polishing the fence so not to block any channel or I/O access *Data_type*: float, optional

`-for_partition`	When specified, it enables the partition fence generation capability. When this option is not specified, the power domain fence generation capability is enabled by default.  *Data_type*: bool, optional
`-help`	Outputs a brief description that includes the type and default information for each `add_fences` parameter. For a detailed description of the command and all of its parameters, use the man command: `man add_fences`
`-hinst {hinst(s)}`	Specifies the list of hierarchical instance names to draw fences. This option can be used if you only want to create a fence for a specific master or clone. You can use wildcards. **Note:** It can not be used for both master and clones. *Data_type*: string, optional
`-inst_group {inst_group(s)}`	Specifies list of instance group names to draw fences. You can use wildcards. *Data_type*: string, optional
`-module {module(s)}`	Specifies list of partition modules to draw fences. This option should be used for master/clone partitions. *Data_type*: string, optional
`-rectangle_shape`	When specified, only rectangle shaped fences are generated. *Data_type*: bool, optional
`-target_density float`	Specifies the target utilization value. *Default*: Core Utilization *Min*: 1e-07 *Max*: 1 *Data_type*: float, optional

Example

- The following command enables the auto-fence generation 2G flow and creates fences for the $PowerDomainNameList group.
  ```
  add_fences -flow 2G -inst_group $PowerDomainNameList
  ```
- The following command enables the auto-fence generation 2G flow and creates fences for specified hierarchical instance(s).
  ```
  add_fences -flow 2G -hinst $hInstNameList -for_partition
  ```

Related Information

- `proto_design`
- Prototyping Methodologies chapter in the *Innovus Stylus Common UI User Guide*

add_interposer_route_block

```
add_interposer_route_block
[-help]
-cell cell_name
-inst clone_name
-location {x y}
-orient {r0 r90 r180 r270 mx my mx90 my90}
```

Adds the clone blocks. It transforms the similar unrouted areas to a clone partition, which repeats the same routing pattern as the routed master block. This command enables you to replicate the routed master block multiple times with different orientations.

Note: Before using the `add_interposer_route_block` command, you must launch another Innovus session for loading the initial DB followed by using `create_interposer_route_block` command to create the routed master block partition by specified box. This is used to generate master block partition and top DB.

Parameters

`-cell cell_name`	Specifies the master cell name that needs to be repeated. *Data_type*: string, required
`-help`	Outputs a brief description that includes the type and default information for each `add_interposer_route_block` parameter. For a detailed description of the command and all of its parameters, use the man command: `man add_interposer_route_block`
`-inst clone_name`	Specifies the clone block name. *Data_type*: list, required
`-location {x y}`	Specifies the clone block's origin location. **Note:** The relative location between clone block's origin and the terms should exactly be the same with the master. Otherwise, the wire of clone blocks will be shift accordingly after assemble design. **Note:** The snapping grid of the created block honors the `floorplan_snap_block_grid` attribute settings. *Data_type*: point, required
`-orient {r0 r90 r180 r270 mx my mx90 my90}`	
	Specifies the orientation of the clone block compared with the master. *Data_type*: enum, required

Examples

- The following command replicates the routing pattern of the master cell (`master_block`) to the clone block (`clone_block_1`) at location x1 y1 at orientation r0.
  ```
  add_interposer_route_block -cell master_block -inst clone_block_1 -location {x1 y1} -orient r0
  ```
- The following command repeats replicating the routing pattern of the master cell (`master_block`) to the clone block (`clone_block_2`) at location x2 y2 at orientation r180.
  ```
  add_interposer_route_block -cell master_block -inst clone_block_2 -location {x2 y2} -orient r180
  ```

Related Information

- `create_interposer_route_block`
- `floorplan_snap_block_grid` attribute
- "Repeating Routing Patterns in Interposer Designs" in the *Innovus Stylus Common UI User Guide*

add_io_fillers

```
add_io_fillers
[-help]
-cells {name_list}
[-fill_any_gap]
[-filler_orient r0 | r90 | r180 | r270 | mx | mx90 | my | my90]
[-from coord]
[-prefix prefix]
[-side {top | bottom | left | right}]
[-to coord]
[-io_ring row_number]
[-use_small_io_height]
[-logic [-derive_connectivity ]]
```

Adds I/O instances in the I/O box. The I/O instances are added between the gap of existing I/O pad instances where the gap is large enough for the I/O instance. Use this command during or after floorplanning.

Note: Before using `add_io_fillers`, run the connect_global_net command to provide global-net-connection rules for supply pins of the added fillers. Without these rules, the built-in design-rule checks of `add_io_fillers` will not be accurate.

Parameters

`-help`	Prints a brief description that includes the type and default information for each `add_io_fillers` parameter. For a detailed description of the command and all of its parameters, use the man command: `man add_io_fillers`
`-cells {name_list}`	Specifies the name list of the I/O filler cells to be added. *Data_type*: list, required
`-derive_connectivity`	Derives connectivity for I/O fillers. *Data_type*: bool, optional
`-fill_any_gap`	Forces the I/O filler instance into a gap even though the gap (clearance) is not large enough. *Default*: If you do not specify this parameter, it does not force a filler instance. *Data_type*: bool, optional
`-filler_orient r0 | r90 | r180 | r270 | mx | mx90 | my | my90`	
	Specifies the orientation value of the filler cell. The orientation value can be r0, r90, r180, r270, mx, mx90, my, or my90. *Default*: r0 *Data_type*: string, optional
`-from coord`	Specifies the starting offset from the edge for adding fillers. The values are in micrometers. **Note:** The `-from` and `-to` parameters are intended to be used with the `-side` parameter to indicate a range. This command is applicable to peripheral I/O only; therefore, it is only one dimensional. *Data_type*: float, optional
`-io_ring row_number`	Specifies the row number for inserting fillers in a multiple I/O row design. *Minimum*: 0 *Maximum*: 1 *Data_type*: int, optional
`-logic`	Enables the creation of I/O filler instances for non-physical I/O cells. The added IO fillers are not marked as physical only cells. They are saved in netlist if user does `write_netlist`. *Data_type*: bool, optional
`-prefix prefix`	Specifies the prefix name for the new I/O filler instances. *Default*: `filler` *Data_type*: string, optional

`-side {top | bottom | left | right}`	
	Specifies the side as top side, bottom side, left side, and right side of the I/O box. *Default*: If you do not specify this parameter, all sides are specified. *Data_type*: string, optional
`-to` *coord*	Specifies the ending offset from the edge for adding fillers. The values are in micrometers. **Note:** The `-from` and `-to` parameters are intended to be used with the `-side` parameter to indicate a range. This command is applicable to peripheral I/O only; therefore, it is only one dimensional. *Data_type*: float, optional
`-use_small_io_height`	Specifies that the filler cell should be inserted between I/O pads only when the smallest I/O pad has the same height as the height of the filler cell. For example, consider that there are I/O pads A, B, C, D, and E. Out of these, A, B, and E are the same height, and C and D have a height double that of A, B, and E. Assume that you want to insert an I/O filler cell F, which has the same height as A, B, and E. If you do not specify the `-use_small_io_height` parameter, the cell F will be inserted between A and B, B and C, C and D, and D and E. However, if you specify the `-use_small_io_height` parameter, the cell F will be inserted between A and B, B and C, and D and E. However, the cell will not be inserted between C and D, because both C and D have a height different from F. *Data_type*: bool, optional

Examples

- The following command adds I/O instances to the I/O pad area on all sides wherever there is space between existing I/O pads. The added I/O pad filler instance names have a prefix PAD8.
  ```
  add_io_fillers -cells PAD8 -prefix PAD8
  ```
- The following command adds I/O instances to the I/O pad area on the top side from 200 micrometers. The added I/O pad filler instance names have a prefix PAD8.
  ```
  add_io_fillers -cells PAD8 -prefix PAD8 -side top -from 200
  ```
- The following command adds I/O instances to the I/O pad area on the left side to 300 micrometers. The added I/O pad filler instance names have a prefix PAD8.
  ```
  add_io_fillers -cells PAD8 -prefix PAD8 -side left -to 300
  ```

Related Information

- `add_area_io_fillers`
- `add_area_io_rows`
- `add_io_insts`
- `add_io_row_fillers`
- `delete_io_fillers`
- `connect_global_net`
- Floorplanning the Design chapter in the *Innovus Stylus Common UI User Guide*

add_io_insts

```
add_io_insts
[-help]
[-cells {name list}]
[-orient orient] -prefix {name list} [-spacing value] [-spread]
{{-ref_inst inst_name [-counter_clockwise ]} |
{-selected [-counter_clockwise ]} |
{-repeat num [-skip num]
[-in_selected | {[-side {n | w | s | e}] [-io_ring num]} | -io_row name]} | -location {x y}}
```

Specifies constraints for insertion of power and ground I/O instances in the I/O ring or in the I/O rows. This command inserts physical IO cells into design following constraints. It is used during floorplanning. If you do not specify any of the following parameters, the command `add_io_insts` adds I/Os for all I/O cells, on all sides and for all I/O rings.

- `-io_ring num`
- `-side`
- `-io_row name`

Parameters

`-cells {name list}`	Specifies one or more cells. Adds one I/O for each cell that was specified. *Data_type*: string, optional
`-counter_clockwise`	Specifies the counter-clockwise direction to add the new I/O instance. *Data_type*: bool, optional
`-help`	Outputs a brief description that includes type and default information for each `add_io_insts` parameter.For a detailed description of the command and all of its parameters, use the man command: `man add_io_insts`
`-in_selected`	Repeatedly inserts new I/O instance(s) within the selected I/O cell(s). *Data_type*: bool, required
`-prefix {name list}`	Specifies the instance name prefix for the new I/O. This must match with the number of cells that are specified. *Data_type*: string, required
`-io_ring num`	Specifies the I/O ring number in which the new I/O instance is added. *Default*: 0 *Data_type*: int, optional
`-io_row name`	For the new I/O row flow, adds I/Os on the specified I/O row. *Data_type*: string, optional
`-location {x y}`	Adds the new I/O instance at the location specified by *x y* coordinates. **Note**: You cannot use this parameter in conjunction with the `-ref_inst` or `-selected` parameters. *Data_type*: point, required
`-orient orient`	Specifies the orientation of the new I/O instance. You can specify one of the following values: `r0`, `r90`, `r180`, `r270`, `mx`, `mx90`, `my`, and `my90`. Use this parameter with `-cells` and `-location` parameters. **Note**: When specifying the `-ref_inst` or `-selected` parameters, the software uses the orientation from those instances. See Orientation Key for more information. *Default*: `r0` *Data_type*: string, optional

`-ref_inst` *inst_name*	Specifies the name of the reference instance and adds the new I/O instance above or below the reference instance in clockwise direction. *Default*: Clockwise direction of the reference instance. *Data_type*: string, optional
`-repeat` *num*	Repeatedly adds new I/O instances every number (num) of cells that you specify. *Default*: Adds instances in the clockwise direction of the reference instance. *Data_type*: int, required
`-selected`	If `-repeat` is specified, this parameter adds the new I/O instance(s) in the selected cell(s) in clockwise direction of the selected instance. If `-repeat` is not specified, this parameter changes the selected cells as referenced cells and adds new I/O instances in the referenced cells, in the default clockwise direction. *Data_type*: bool, required
`-side {n | w | s | e}`	Specifies the side on which the new I/O instance is added. *Default*: `n` *Data_type*: string, optional
`-skip` *num*	Skips the number (num) of cells specified, before adding the first I/O instance. This happens only when `-repeat` num is specified. *Data_type*: int, optional
`-spacing` *value*	Specifies the spacing value, in microns, between the I/O instances. *Data_type*: float, optional
`-spread`	Evenly replaces all the I/O cells on the same row where the new I/O instance was added. This prevents overlapping of I/O cells. If you do not specify this parameter, the command still adds the new I/Os even if overlapping occurs. **Note**: The `add_io_insts` command works for both the I/O flows, the old flow and the new row based pad placement flow. In the new flow, each time you add a new I/O instance, the I/Os on the I/O rows with the existing constraints are replaced. In this situation, if all the existing I/Os do not have any specific constraint defined, then the replacement of the I/Os also resolves overlapping each time when the I/O cell is added, evading the usage of `-spread` option. *Data_type*: bool, optional

Example

- The following command adds new I/O instance il to cell BUMPCELL at lower-left x location 100 and lower-left y location 100:
  ```
  add_io_insts -prefix il -cells BUMPCELL -location 100 100
  ```
- The following command inserts the I/O instance newIo for the cell PVDD on the West side of the I/O box, skipping 3 I/O's (from below) on the West side and repeating the insertion of I/O's every 2 instances.
  ```
  add_io_insts -cells PVDD -repeat 2 -prefix newIo -side w -skip 3 -spread
  ```

Related Information

- `add_area_io_fillers`
- `add_area_io_rows`
- `add_io_fillers`
- `add_io_row_fillers`
- `delete_io_insts`
- Floorplanning the Design chapter in the *Innovus Stylus Common UI User Guide*

add_io_row_fillers

```
add_io_row_fillers
[-help]
-cells {filler_cell_name_list ...}
[-logic]
[-prefix prefix]
[[-io_row name | -power_domain power_domain_name]
[-filler_orient r0 | r90 | r180 | r270 | mx | mx90 | my | my90]]
[-from coord] [-to coord]
[-fill_any_gap]
[-use_small_io_height]
[-ignore_site_type]
```

Adds I/O filler cells in the I/O rows. The I/O filler cells are added between the gap of existing I/O pad instances where the gap is large enough for the I/O cell. Use this command during or after floorplanning.

Parameters

`-cells {`*`filler_cell_name_list ...`*`}`	
	Specifies the name of the filler cell(s) to add in the I/O row. *Data_type*: string, required
`-fill_any_gap`	Forces the I/O filler instance into a gap even though the gap (clearance) is not large enough. *Default*: If you do not specify this parameter, it does not force a filler instance. *Data_type*: bool, optional
`-filler_orient r0 | r90 | r180 | r270 | mx | mx90 | my | my90`	
	Specifies the orientation value of the filler cell. The orientation value can be r0, r90, r180, r270, mx, mx90, my, or my90. Default: `r0` *Data_type*: string, optional
`-from` *`coord`*	Specifies the coordinates of a specified starting range where I/O filler cells are added. The values are in micrometers. *Data_type*: float, optional
`-help`	Outputs a brief description that includes type and default information for each `add_io_row_fillers` parameter. For a detailed description of the command and all of its parameters, use the man command: `man add_io_row_fillers`
`-ignore_site_type`	Ignores the I/O row site type. *Data_type*: bool, optional
`-io_row` *`name`*	Specifies the name of the I/O row to add the filler cell. *Data_type*: string, optional
`-logic`	Adds logical I/O fillers to net list in top module. *Data_type*: bool, optional
`-power_domain` *`power_domain_name`*	
	Adds I/O filler instances to the I/O rows associated with the specified power domain. *Data_type*: string, optional
`-prefix` *`prefix`*	Specifies the prefix name to be appended for the added I/O row filler cell. *Data_type*: string, optional
`-to` *`coord`*	Specifies the coordinates of a specified ending range where I/O row filler cells are added. The values are in micrometers. *Data_type*: float, optional
`-use_small_io_height`	Specifies that the filler cell should be inserted between I/O pads only when the smallest I/O pad has the same height as the height of the filler cell. For example, consider that there are I/O pads A, B, C, D, and E. Out of these, A, B, and E are the same height, and C and D have a height double that of A, B, and E. Assume that you want to insert an I/O filler cell F, which has the same height as A, B, and E. If you do not specify the `-use_small_io_height` parameter, the cell F will be inserted between A and B, B and C, C and D, and D and E. However, if you specify the `-use_small_io_height` parameter, the cell F will be inserted between A and B, B and C, and D and E. However, the cell will not be inserted between C and D, because both C and D have a height different from F. *Data_type*: bool, optional

Example

- The following command adds I/O instances to the I/O pad area on all sides wherever there is space between existing I/O pads. The added I/O pad filler instance names have a prefix PAD8.

```
add_io_row_fillers -cells PAD8 -prefix PAD8
```

Related Information

- `add_area_io_fillers`
- `add_area_io_rows`
- `add_io_fillers`
- `add_io_insts`
- `delete_io_row_fillers`
- Floorplanning the Design chapter in the *Innovus Stylus Common UI User Guide*
  - Performing I/O Row Based Pad Placement

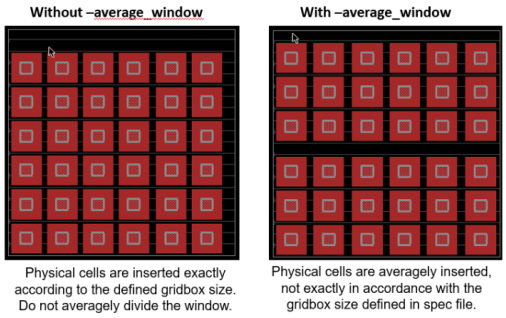
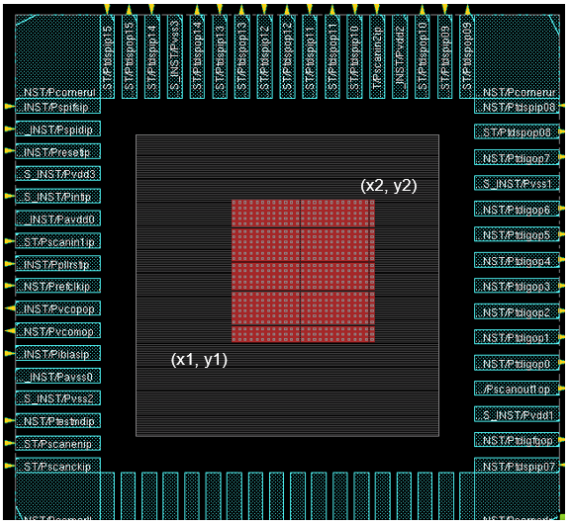
add_physical_cell

```
add_physical_cell
[-help]
[-cell string]
[-generate_template_only string]
[-input_file filename]
[-insert_row_area_boundary]
[-average_window | { [-insert_near_boundary ] [-box {x1 y1 x2 y2}] [-minimal_number_of_tcd integer]}]
[-verify_window {x y}[ [-verify_only ][-x_step double] [-y_step double] ] ]
```

Provides a flexible way to insert and check physical cells. It can be used to assign a specific area for physical insertion, not only the die area.

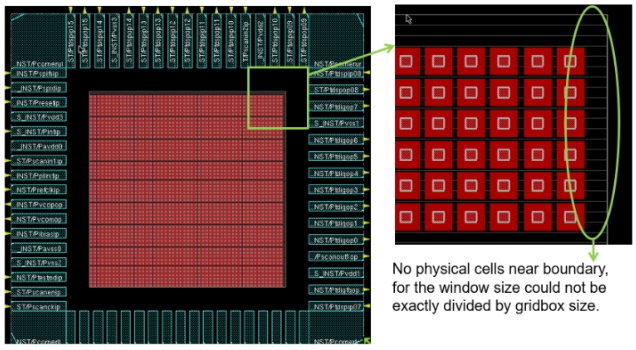
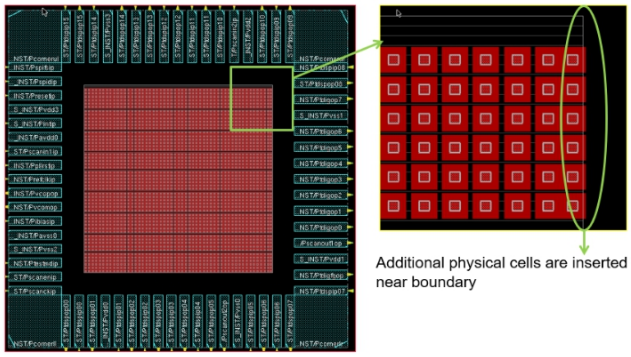
Note: The `add_physical_cell` command is suitable for inserting physical cells as a pattern. The class of these cells must be block and not core. The `add_physical_cell` command is not when the class of the physical cells is core and there is no requirement for inserting cells in a pattern.

Parameters

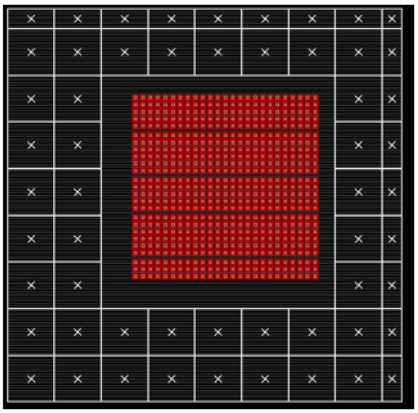
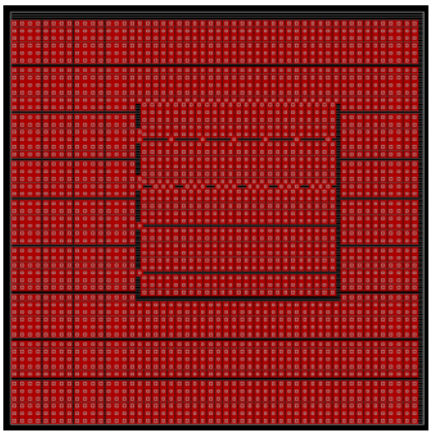
`-average_window`	When specified, it changes the window calculation approach to average. The size of windows for inserting cell is the average of the core box. The insertion will be done not exactly in accordance with the size defined in spec file.  *Data_type*: bool, optional
`-box {x1 y1 x2 y2}`	Specify the bounding box of the physical cell in `x1 y1 x2 y2`. For example, the following command inserts the cell DTCDCELL in the (*x1 y1 x2 y2)* box. `add_physical_cell -cell DTCDCELL -input_file SPECFILE -box {x1 y1 x2 y2}`  *Data_type*: box, optional
`-cell string`	Specify the physical cell name. *Data_type*: string, optional

`-generate_template_only` *string*	Generates a template file to guide how to define spec file for DTCD, ICOVL and LinerCell. *Data_type*: string, optional
`-help`	Outputs a brief description that includes the type and default information for each `add_physical_cell` parameter. For a detailed description of the command and all of its parameters, use the man command: `man add_physical_cell`

`-input_file` *filename*	Specify the spec file for physical cells.
	Note: The inserted physical cells must be specified in a spec file. This file specifies the information related to the dimensions and spacing physical cells. The tool will insert physical cells according to these specifications.
	Here is an example of a spec file:
	`**********`
	`DTCD Cell cell_name`
	`WinSizeX  # x dimension of DTCD insertion grid box`
	`WinSizeY  # y dimension of DTCD insertion grid box`
	`MacroSpacing  # spacing between DTCD cell and macro`
	`DTCDSpacing  # spacing between DTCD cells`
	`PlaceBlockSizeX  # x dimension of padded placement blockage around DTCD cell`
	`PlaceBlockSizeY  # y dimension of padded placement blockage around DTCD cell`
	`RouteBlockSizeX  # x dimension of padded routing blockage around DTCD cell`
	`RouteBlockSizeY  # y dimension of padded routing blockage around DTCD cell`
	`End`
	`**********`
	Data_type: string, optional

`-insert_near_boundary`	Inserts physical cells near boundary.
	Note: When the window size is not exactly divided by gridbox size, you may not get physical cells near boundary.
	For example:
	`add_physical_cell -cell DTCDCELL -input_file SPECFILE`
	 
	No physical cells near boundary, for the window size could not be exactly divided by gridbox size.
	`add_physical_cell -cell DTCDcell -input_file SPECFILE -insert_near_boundary`
	 
	Additional physical cells are inserted near boundary
	Data_type: bool, optional

`-insert_row_area_boundary`	Inserts LinerCell around row area boundary. *Data_type*: bool, optional
`-minimal_number_of_tcd` *integer*	Specifies the the minimal number of DTCD that must be inserted. If the current windows size definition (*WinSizeX* and *WinSizeY*) cannot fit this requirement, the window size is shrunk automatically. *Data_type*: int, optional

`-verify_only`	Only verifies the physical cells to inserted. It does not actually insert them.
	Note: The `-verify_window`, `-x_step`, and `-y_step` parameters are used to specify the size of the window to verify.
	For example:
	`add_physical_cell -cell DTCDCELL -input_file SPECFILE -verify_window 100 100 -x_step 100 -y_step 100 -verify_only`
	 
	Only verify the physical cell in gridbox, do not insert.
	`add_physical_cell -cell DTCDCELL -input_file SPECFILE -verify_window 100 100 -x_step 100 -y_step 100`
	 
	Verify and insert the physical cells
	Data_type: bool, optional

`-verify_window {`*x y*`}`	Specifies the size of window for verifying the physical cells to be inserted. *Data_type*: point, optional
`-x_step` *double*	Specifies the x step of the verify window. *Data_type*: double, optional

`-y_step` *double*	Specifies the y step of the verify window. *Data_type*: double, optional

Example

- The following command inserts the cell DTCDCELL in the *(x1 y1 x2 y2)* box based on the spec specified in the SPECFILE.
  ```
  add_physical_cell -cell DTCDCELL -input_file SPECFILE -box {x1 y1 x2 y2}
  ```
- The following command only verifies the cell DTCDCELL in the grid box. It does not insert the cell.
  ```
  add_physical_cell -cell DTCDCELL -input_file SPECFILE -verify_window 100 100 -x_step 100 -y_step 100 -verify_only
  ```

Related Information

- Floorplanning the Design chapter in the *Innovus Stylus Common UI User Guide*

align_selected

```
align_selected
[-help]
-side {right | left | center | top | bottom | middle}
[-refer_to_first]
[-mix]
```

Aligns objects (instances, modules, or blockages) vertically or horizontally in the floorplan. Use this command after selecting two or more objects in the design display area.

Parameters

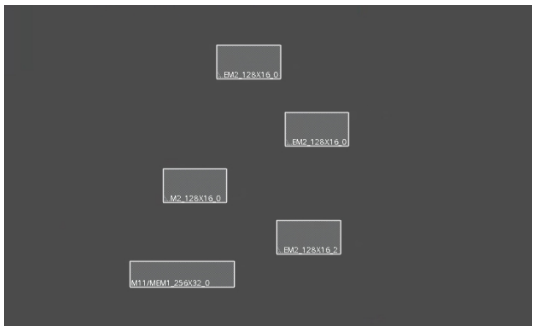
`-help`	Outputs a brief description that includes the type and default information for each `align_selected` parameter. For a detailed description of the command and all of its parameters, use the man command: `man align_selected`
`-mix`	Considers all the selected objects together while aligning them. *Data_type:* bool, optional
`-refer_to_first`	If you specify this option, the first selected object is the reference object. *Data_type:* bool, optional
`-side {right | left | center | top | bottom | middle}`	
	Specifies the vertical and horizontal alignment for the selected objects. Choose one of the following: • `right`: Specifies vertical right alignment. • `left`: Specifies vertical left alignment. • `center`: Specifies vertical center alignment. • `top`: Specifies horizontal top alignment. • `bottom`: Specifies horizontal bottom alignment. • `middle`: Specifies horizontal middle alignment. *Data_type:* string, required

Example

- The following command aligns the selected objects (Block) vertically to the left:

```
align_selected -side left
```

Before:

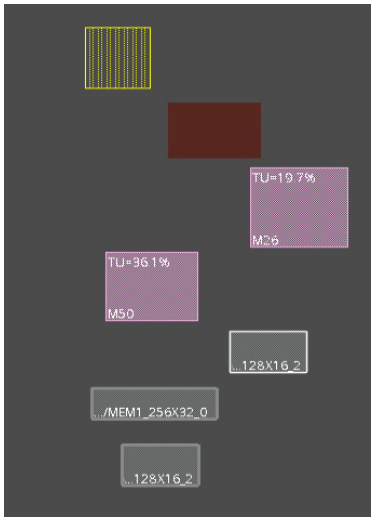


After:

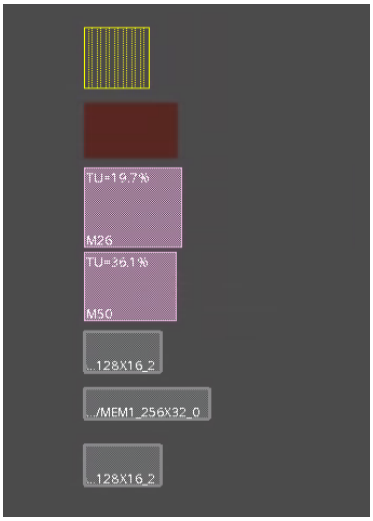
- The following command aligns the selected objects (Blockage、Module、Block) vertically to the left：

```
align_selected -side left -mix
```

Before:



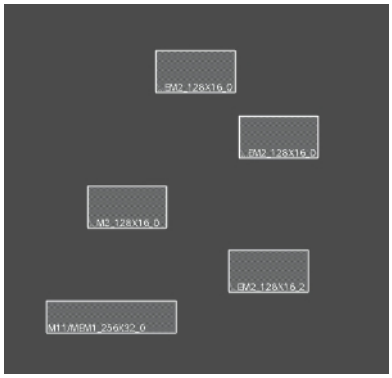
After:



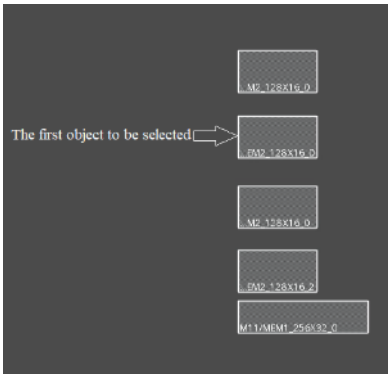
- The following command aligns the selected objects (Block) vertically to the left：

```
alignObject -side left -referToFirst
```

Before：



After：



Related Information

- `space_selected`
- `flip_or_rotate_obj`
- Floorplanning the Design chapter in the *Innovus Stylus Common UI User Guide*

check_floorplan

```
check_floorplan
[-help]
[-out_file file_name]
[-report_density]
```

Checks the quality of the floorplan to detect potential problems before the design is passed on to other tools. This highlights the cells in the design display area with violation markers, where applicable. Use this command after specifying the floorplan data or running an initial floorplan. Run check_floorplan on your final floorplan file.

Notes:

- The `floorplan` category attributes that you define are used automatically whenever you run the `check_floorplan` command.
- Innovus honors the FinFET grid and automatically checks for the FinFET definition. If it detects a FinFET definition it enables support and sets the placement grid as FinFET grid by default. Since the height of all placeable objects must be a multiple of FinFET grid and the origin of these objects must be snapped to the FinFET grid. The check_floorplan command checks if all the supported objects (die box, core box, standard cell rows, I/O rows, blocks, IO pads, and standard cells) are aligned with FinFET grid and reports DRC violations.
- The `check_floorplan` command can check if the row's orientation is correct according to related tech site's SYMMETRY definition. It flags when Y SYMMETRY is missing from techlef CORE site definition.
- The `check_floorplan` command checks the pin snapping result.
  - For min width pins, it checks whether the pin center is snapped to the routing track with the same color.
  - For fat pins, it checks whether the pin center is snapped to the routing track and decides if the pin center should snap to the same or opposite colored routing track.
- The `check_floorplan` command checks if power stripes are on routing track for 10nm DPT layers (Metal1, Metal2, Metal3). It also checks power vias upper and lower metal shapes if they are not entirely covered by other power stripes on the same layers. Additionally it also checks if multi-height rows conflict with PG rails. In case of a conflict, a violation is reported.
- You can use the `check_floorplan` command to check if there are any existing overlaps between group boxes and power domain boxes as the box for a group should not physically overlap with the box for a power domain.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `check_floorplan` parameter. For a detailed description of the command and all of its parameters, use the man command: `man check_floorplan`
`-out_file file_name`	Outputs detailed information for the specified checks. *Data_type:* string, optional
`-report_density`	Reports target utilization (TU) and effective utilization (EU) for the entire design, fences, and regions (partitions, power domains, and regular fences). *Data_type:* bool, optional

Example

- The following command checks the floorplan and writes the detailed information to the details.txt file. It also reports the final core utilization:
  ```
  check_floorplan -out_file details.txt -report_density
  ```

Related Information

- `floorplan` Category Attributes
- `check_design`
- `check_floorplan_space`
- `create_floorplan`
- Floorplanning the Design chapter in the *Innovus User Guide*

check_floorplan_space

```
check_floorplan_space
[-help]
[-out_file filename]
[-in_file filename | -track_utilization layer_id value layer_id value ...]
[-clear_markers]
```

Checks the floorplan for spacing rule violations and saves the violation information in a report. A marker is displayed at each violation spot. You can use the command at any point in the design flow.

Parameters

`-clear_markers`	Clears all space violation markers. *Data_type*: bool, optional
`-in_file filename`	Specifies the file that contains the spacing rules. A constraint file is a text file that can contain a list of seeds to use when generating fences and module guides, and basic relative, spacing and orientation constraints to follow during macro placement. *Data_type*: string, optional
`-help`	Outputs a brief description that includes type and default information for each `check_floorplan_space` parameter. For a detailed description of the command and all of its parameters, use the man command: `man check_floorplan_space`
`-out_file filename`	Writes the spacing rule violation information to the specified file. *Data_type*: string, optional
`-track_utilization layer_id value layer_id value ...`	
	Specifies the track utilization value on each routing layer from bottom to top, for example, `Metal1 value Metal2 value`. If the layer track utilization value is not specified, the default value is used. **Note:** The `check_floorplan_space` command supports the model based space checker. It takes user provided track utilization numbers and technology information stored in the LEF file to estimate required minimum channel width between placed macros. This estimated minimum channel width (demand) is then compared to actual channel width (supply) in floorplan. A violation marker is displayed in GUI over the channel in which the supply is less than the demand. *Data_type*: list, optional

Related Information

- `check_floorplan`
- `check_design`
- Floorplan Menu chapter in the *Innovus Stylus Common UI Menu Reference*
  - Check Floorplan Space Rule

check_macro_lower_left_on_track

```
check_macro_lower_left_on_track
[-help]
-use_layer_offset
-use_m2_m3_track
```

By default, checks whether the lower left coordinates (x1, y1) of hard macros are at the intersection of M1/M2 tracks, and reports an error in Innovus if they are not. This is based on the assumption that Ml1 and M2 are both specified in opposite directions in a 65nm process. You can use this command after the floorplan is ready and want to make sure hard macro is on tracks.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `check_macro_lower_left_on_track` parameter. For a detailed description of the command and all of its parameters, use the man command: `man check_macro_lower_left_on_track`
`-use_layer_offset`	Checks pin alignment in a block against the layer offset of the design library. *Data_type*: bool, optional
`-use_m2_m3_track`	Checks the location of the macro against M2/M3 tracks if both M1 and M2 are specified in the same horizontal direction in a 40nm process. *Data_type*: bool, optional

Example

- The following command checks whether the lower left coordinates (llx, lly) of hard macros are at the intersection of Metal1/Metal2 tracks:
  ```
  check_macro_lower_left_on_track -use_layer_offset -use_m2_m3_track
  ```

Related Information

- `check_macro_place_constraint`
- `set_macro_place_constraint`
- Floorplanning the Design chapter in the *Innovus Stylus Common UI User Guide*

check_macro_place_constraint

```
check_macro_place_constraint
[-help]
```

Checks the constraints of macros which will be honored during the macro placer. To define the constraints of macros, use the `set_macro_place_constraint` command.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `check_macro_place_constraint` parameter. For a detailed description of the command and all of its parameters, use the man command: `man check_macro_place_constraint`

Related Information

- `create_pg_model_for_macro_place`
- `get_macro_place_constraint`
- `reset_macro_place_constraint`
- `set_macro_place_constraint`
- `write_macro_place_constraint`
- Floorplanning the Design chapter in the *Innovus Stylus Common UI User Guide*
- Using the Mixed Placer chapter in the *Innovus Stylus Common UI User Guide*

check_ufc

```
check_ufc
[-help]
fileName
[-include_all_rules]
[-report_file string]
[-report_unplaced_block]
[-selected_rules string]
```

Checks the floorplan based on the user specified UFC rules defined in the UFC file and reports violations. These violations can be seen in the *Violation Browser*.

Note: The Unified Floorplan Constraints (UFC) methodology is a powerful Innovus capability that can define several rules on floorplan objects, check specified rules, and report the violations with rule name as well as severity. The violations can be shown in the violation browser. UFC also provides the ability to automatically fix specific violations. It can add routing blockages, snap macro location, adjust core shape to fix corresponding violations. In addition, UFC also provides the ability to exclude rules from checking for specified objects.

Usage Model:

- Load the Design.
- Create a UFC file with floorplan rules.
- Run UFC checking. (`check_ufc`)
- Fix reported violations. (`fix_floorplan`)

Note: The check_ufc command honors the `floorplan_no_cut_row` attribute that specifies if rows are not required to be cut before operations. With `floorplan_no_cut_row false` or not specified, the `check_ufc` command will work based on the current row shape. If there are rules for rows in the UFC file and the rows are not cut before, there will be many wrong violations reported out. So the rows for objects like insts, place blockages are expected to cut out before using the `check_ufc` command. If rows are not cut before the check_ufc command is run, with `floorplan_no_cut_row true`, the tool identifies the rows have not been cut before. It calls the `split_row` command to get a cut row shape during the `check_ufc` command. The `split_row` command cuts the rows for insts, hinsts, place blockages, route blockages, power domains, groups, or bumps by default. The check_ufc command will work based on the cut row shape. The actual row shape is not changed after these operations. You can still get the expected check result for row's rule without running the `split_row` before.

Parameters

fileName	Specifies the name of file containing the rules or constraints. *Data_type*: string, required
-help	Prints a brief description that includes type and default information for each `check_ufc` parameter. For a detailed description of the command and all of its parameters, use the man command: `man check_ufc`
-include_all_rules	Enables the tool to report the summary of violations for all rules. *Data_type*: bool, optional
-report_file *string*	Specifies the name of the file that saves the report summary. *Data_type*: string, optional
-report_unplaced_block	Enables the tool to show the number of unplaced macros in summary. *Data_type*: bool, optional

-selected_rules *string*	Specifies that only the selected rules will be checked.
	Note: The `-selected_rules` parameter supports the variable format for specifying the rule list.
	For example, you can define a rule in the UFC file and then use the `check_ufc` command to check the rule.
	`set sanity_rules 'A C'`
	`check_ufc file.ufc -selected_rules $sanity_rules`
	Here, rule A and C will be checked
	Data_type: string, optional

Example

- The following command checks the floorplan based on the *DMTB_ENC.ufc* file that contains the user specified UFC rules:

```
check_ufc DMTB_ENC.ufc
```

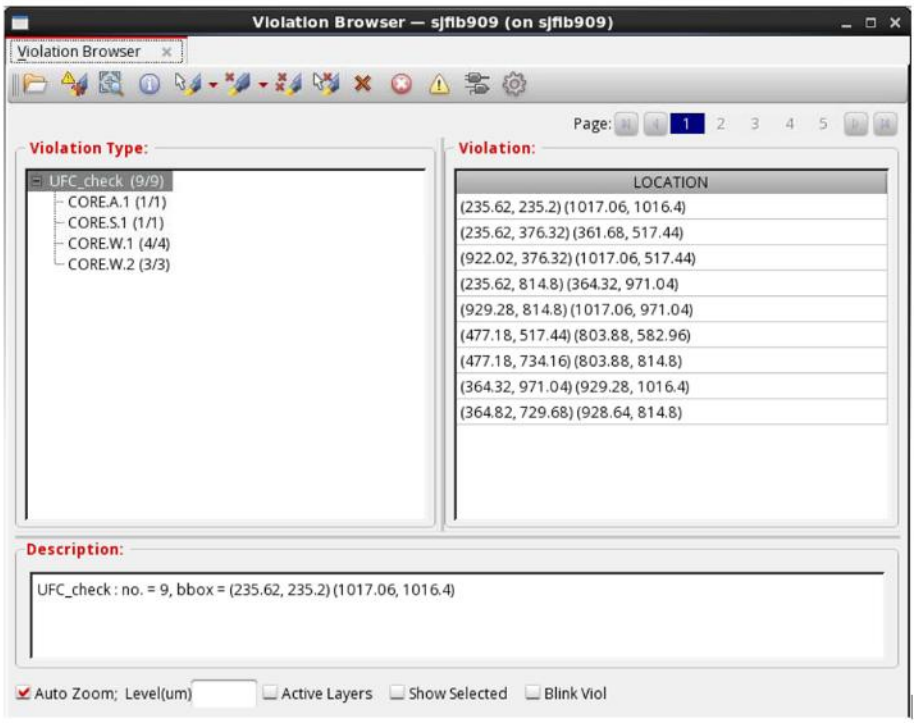
The input UFC file is like this:

```
set_area_rule -name CORE.A.1 -obj [available_sites] {.area < 500} \
  -description {The area of available sites should be lesser than 500}
set_width_rule -name CORE.W.1 -obj [available_sites] -direction horizontal -type simple {.width > 200} \
  -description {The horizontal simple width of available sites must be larger than 200}
set_width_rule -name CORE.W.2 -obj [available_sites] -direction vertical -type simple {.width > 100} \
  -description {The vertical simple width of available sites must be larger than 100}
set_spacing_rule -name CORE.S.1 -obj [base_cells -if {.name ==ram_256x16A}] -ref [base_cells -if {.name == rom_512x16A}] -direction
vertical {on_grid (.spacing-0.1,0.2)} \
  -description {The vertical spacing between macro ram_256x16A and rom_512x16A must be 0.2*n + 0.1}
set_spacing_rule -name CORE.S.2 -obj [base_cells -if {.name ==rom_512x16A}] -ref [base_cells -if {.name == ram_128x16A}] -direction
vertical {.spacing > 10} \
  -description {The vertical spacing between macro rom_512x16A and ram_128x16A must larger than 10}
```

Following is an example of report summary of UFC checking after running the check_ufc command. The violations can be seen in the Violation Browser.

```
-------------------------Summary of UFC Checking ---------------------------------------
RuleName Count  Description                                                          ViolatedObjects
........ .....  ............................                                         ..........................
CORE.A.1   1    The area of available sites should be lesser than 500                available_sites
CORE.S.1   1    The vertical spacing between macro ram_256x16A and rom_512x16A must be 0.2*n + 0.1
 DTMF_INST/RAM_256x16_TEST_INST/RAM_256x16_INST
CORE.W.1   4    The horizontal simple width of available sites must be larger than 200   available_sites
CORE.W.2   3    The vertical simple width of available sites must be larger than 100     available_sites
```

The violations can be seen in the Violation Browser.



- The following command checks the floorplan based on the *DMTB_ENC.ufc* file and includes all the rules:

```
check_ufc -include_all_rules
```

Following is an example of report summary:

```
-------------------------Summary of UFC Checking ---------------------------------------
RuleName Count  Description                                                          ViolatedObjects
........ .....  ............................                                         ........................
CORE.A.1   1    The area of available sites should be lesser than 500                available_sites
CORE.S.1   1    The vertical spacing between macro ram_256x16A and rom_512x16A must be 0.2*n + 0.1
 DTMF_INST/RAM_256x16_TEST_INST/RAM_256x16_INST
CORE.W.1   4    The horizontal simple width of available sites must be larger than 200    available_sites
CORE.W.2   3    The vertical simple width of available sites must be larger than 100     available_sites
CORE.S.2   0    The vertical spacing between macro rom_512x16A and ram_128x16A must be larger then 10
```

- The following command checks the floorplan based on the *DMTB_ENC.ufc* file and includes all the rules:
  ```
  check_ufc -include_all_rules -report_unplaced_block
  ```

  Following is an example of report summary:

  ```
  -------------------------Summary of UFC Checking ---------------------------------------
  RuleName Count  Description                                                          ViolatedObjects
  ........ .....  ............................                                         ........................
  CORE.A.1   1    The area of available sites should be lesser than 500                available_sites
  CORE.S.1   1    The vertical spacing between macro ram_256x16A and rom_512x16A must be 0.2*n + 0.1
   DTMF_INST/RAM_256x16_TEST_INST/RAM_256x16_INST
  CORE.W.1   4    The horizontal simple width of available sites must be larger than 200    available_sites
  CORE.W.2   3    The vertical simple width of available sites must be larger than 100     available_sites
  CORE.S.2   0    The vertical spacing between macro rom_512x16A and ram_128x16A must be larger then 10
  **WARN: (IMPFP - 10100): There are 1 unplaced blocks. { DTMF_INST/PLLCLK_INST}. Tool doesn't check and fix violations for these blocks.
  ```

Related Information

- fix_floorplan
- Floorplanning the Design chapter in the *Innovus Stylus Common UI User Guide*

convert_fence_to_lef

```
convert_fence_to_lef
[-help]
hinst
[partition_cell]
```

Converts a specified hierarchical instance or partition name to a LEF block. Use this command after floorplanning the design.

Note: This command runs `commit_partitions` and `write_partitions` for the specified partition as if it was at the top-level. This will overwrite any partition or hierarchical instance that is specified in the .lef file in the top partitioned subdirectory. In addition, the *ptnname.def* file will overwrite the partition's *ptn.def* subdirectory file.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `convert_fence_to_lef` parameter. For a detailed description of the command and all of its parameters, use the man command: `man convert_fence_to_lef`
hinst	Specifies the hierarchical instance cell name. *Data_type*: string, required
partition_cell	Specifies the partition name. If this is not specified, the command uses the hierarchical instance cell name as the partition name. *Data_type*: string, optional

Example

The following command converts the results_conv hierarchical instance to a LEF block.

```
convert_fence_to_lef results_conv
```

Related Information

- `commit_partitions`
- `write_partitions`
- Floorplanning the Design chapter in the *Innovus Stylus Common UI User Guide*

copy_obj

```
copy_obj
[-help]
```

Copies the selected placement or routing blockages. Use the `paste_obj` command to paste the copied object to the required location. You can use the `copy_obj` command any time after floorplanning the design. You can also copy selected placement and routing blockages from the GUI by using the Copy tool. To access the Copy tool, choose *Edit - Copy* from the menu bar or press `F3` after clicking the *Copy* widget ( ) on the toolbar.

Note: The `copy_obj` command currently works only for placement and routing blockages. Use the `edit_copy` command to copy wires or vias.

Parameters

`-help`	Prints a brief description of the `copy_obj` command. For a detailed description of the command, use the man command: `man copy_obj`

Related Information

- `paste_obj`
- `edit_copy`
- `move_obj`
- Floorplanning the Design chapter in the *Innovus Stylus Common UI User Guide*

create_boundary_constraint

```
create_boundary_constraint
[-help]
[-floating]
-type {fence region guide cluster}
{-hinst hinst_name | -group group_name}
[-rects {{x1 y1 x2 y2} ...} | -polygon {{x1 y1} {x2 y2} ...}]
```

Creates a boundary constraint for an hinst or a group. Use this command after importing the design.

Parameters

`-floating`	Creates a floating boundary constraint. **Note:** It does not support rectilinear shapes. **Note:** You can use the `-floating` parameter for the `fence` or `region` type of instance groups only. The floating region/fence constraint allows the entire region/fence to move as a cluster across the core area. *Data_type*: bool, optional
`-group group_name`	Specifies the name of the group for which to add the boundary constraint. *Data_type*: (group), optional
`-help`	Outputs a brief description that includes the type and default information for each `create_boundary_constraint` parameter. For a detailed description of the command and all of its parameters, use the man command: `man create_boundary_constraint`
`-hinst hinst_name`	Specifies the name of the hinst for which to add the boundary constraint. *Data_type*: (hinst), optional
`-polygon {{x1 y1} {x2 y2} ...}`	Specifies the vertices of the polygon boundary, in microns. *Data_type*: (point)+, optional
`-rects {{x1 y1 x2 y2} ...}`	Specifies the vertices of the rectangular boundary, in microns. • *x1*: Specifies the lower left x coordinate of the boundary. • *y1*: Specifies the lower left y coordinate of the boundary. • *x2*: Specifies the upper right x coordinate of the boundary. • *y2*: Specifies the upper right y coordinate of the boundary. *Data_type*: (rect)+, optional
`-type {fence region guide cluster}`	Specify the type of the boundary constraint to create. You can specify: • `fence`: Specify to create a fence. • `region`: Specify to create a region. • `guide`: Specify to create a guide. • `cluster`: Specify to create a cluster. **Note:** A cluster (also called softGuide) is similar to a guide constraint except there are no fixed locations. This provides stronger grouping for the instances under the same soft guide. *Data_type*: enum, optional

Example

- The following command creates a fence constraint for hierarchical instance SH17 at the specified location:
  ```
  create_boundary_constraint -hinst SH17 -type fence -rects {100 100 4898 4898}
  ```
- The following command creates a guide constraint for group adder1 at the specified location:
  ```
  create_boundary_constraint -group adder1 -type guide -rects {834.2000 1012.0000 2498.4000 3087.0000}
  ```
- The following command creates a cluster constraint for a hierarchical instance:
  ```
  create_boundary_constraint -hinst top/sub_module1/sub_sub_module2 -type cluster
  ```

Related Information

- Floorplanning the Design chapter in the *Innovus Stylus Common UI User Guide*
  - Module Constraint Types

create_bump_grid

```
create_bump_grid
[-help]
-grid_name string
-layers string
-origin {x y}
-pitch {x y}
```

Creates a bump grid. A bump grid is a grid for bump/TSV placement that helps in guiding bump placement and bump alignment.

Note:

- When you use the `write_partitions` command, the tool recalculates the offset so that a bump can be aligned after assemble.
- The write_floorplan command saves the bump grid definition into *.fp file.
- The read_floorplan command loads the bump grid file while reading the design.

Parameters

`-grid_name string`	Specifies the bump grid name. *Data_type*: string, required
`-help`	Outputs a brief description that includes the type and default information for each `create_bump_grid` parameter. For a detailed description of the command and all of its parameters, use the man command: `man create_bump_grid`
`-layers string`	Specifies the layer names. You can specify the layer of a relative bump pin for this option. *Data_type*: string required
`-origin {x y}`	Specifies the coordinates of the origin of the bump grid. *Data_type*: point, required
`-pitch {x y}`	Specifies the spacing between two bumps. The first value (x) defines the horizontal spacing and the second value (y) defines the vertical spacing. *Data_type*: point, required

Example

- The following command creates a bump grid named myGridFront_9x9:

```
create_bump_grid -grid_name myGridFront_9x9 -layers {UBM UBM1} -origin {50 50} -pitch {9 9}
```

Related Information

- `delete_bump_grid`
- Floorplanning the Design chapter in the *Innovus Stylus Common UI User Guide*

create_exclusive_groups

```
create_exclusive_groups
[-help]
[group_name]
[-gap double]
[-min_gap double]
[-exclusive_groups string | -mutually_exclusive_groups string]
```

Creates exclusive regions. Exclusive regions specified for a particular group cannot be placed/overlapped with it but they can overlap on each other/ any other region in the core area.

Sample Script

Following is a sample script to create exclusive groups:

```
create_group $group_name -type region
update_group -name $group_name -add -objs insts
create_group $group_name1 -type region
update_group -name $group_name1 -add -objs insts
create_exclusive_groups $group_name -exclusive_groups {$groups_name1}
```

Parameters

`-exclusive_groups string`	Specifies the groups/regions that cannot be placed/overlapped with the specified group name. For Example: If regions B,C,D are defined as exclusive regions to A. Instances from B,C,D can never be placed on A. Other regions which are not defined exclusive to region A can be placed or overlapped with region A. *Data_type*: string, optional
`-gap double`	Specifies the value of the gap, in microns, that is maintained among the exclusive groups. *Default*: 5 (microns) *Data_type*: double, optional
`group_name`	Specifies the name of the created group. *Data_type*: string, optional
`-help`	Outputs a brief description that includes the type and default information for each `create_exclusive_groups` parameter. For a detailed description of the command and all of its parameters, use the man command: `man create_exclusive_groups`
`-min_gap double`	Specifies the value of the minimum gap, in microns, that is maintained among exclusive groups. *Data_type*: double, optional
`-mutually_exclusive_groups string`	Specifies the groups which need to be excluded with each other. You can independently use this option to specify and create multiple mutually exclusive groups. **Note:** If a group has already been defined as an exclusive groups, then a second setting for this group using the `create_exclusive_groups` command will give display an error that suggests you to reset the exclusive groups attributes to default value. If you want to create mutually exclusive groups, then reset the exclusive groups attributes using the `delete_exclusive_groups` command and then create the mutually exclusive groups using the `create_exclusive_goups -mutually_exclusive_groups` command. *Data_type*: string, optional

Examples

- The following commands define `gr_300` as an exclusive region to gr_200:

```
create_group gr_200 -type region
update_group gr_200 DTMF_INST/TDSP_CORE_INST/EXECUTE_INST/acc_reg*
create_group gr_300 -type region
update_group gr_300 DTMF_INST/TDSP_CORE_INST/EXECUTE_INST/top_reg_*
create_exclusive_groups gr_200 -exclusive_groups gr_300
```

- The following commands defines instance groups A, B, C to be mutually exclusive groups with a space of 10 microns between each other.

```
create_exclusive_groups -mutually_exclusive_groups {A B C} -gap 10
```

Related Information

- `create_group`
- `update_group`
- `delete_exclusive_groups`
- Floorplanning the Design chapter in the *Innovus User Guide*

create_floorplan

```
create_floorplan
[-help]
[-core_margins_by {io | die}]
[-die_size_by_io_height {min |max}]
[-flip {f | s | n}]
[-floorplan_origin {center | llcorner}]
[-match_to_site]
[-no_resize]
[-no_snap_to_grid]
[-overlap_same_site_row]
{-box_size {die_box io_box core_box} |
-core_size {w h left bottom right top} |
-die_size {w h left bottom right top} |
-core_density_size {aspect_ratio [row_density [left bottom right top]]} |
-stdcell_density_size {aspect_ratio [std_cell_density [left bottom right top]]} |
-keep_shape_density util}
[-site name | -site_only name]
```

Specifies the floorplan dimensions by size; or by die, I/O, or core coordinates. The `create_floorplan` command initializes the floorplan and calls the `add_tracks` command to create new routing tracks. By default the `add_tracks` command calculates the optimum spacing between tracks and ignores the pitch values in the tech LEF. When you use the *Specify Floorplan* form (or specify the floorplan through the floorplan command), the floorplan is resized automatically - relatively floorplan constraints are automatically derived on the fly for blocks, fixed standard cells, fixed pre-routes, and blockages. The floorplan is linearly adjusted as follows:

- Spacing among blocks is evenly adjusted
- The size of the modules and black boxes is evenly adjusted.
- The fixed pre-routes and I/Os are automatically adjusted.

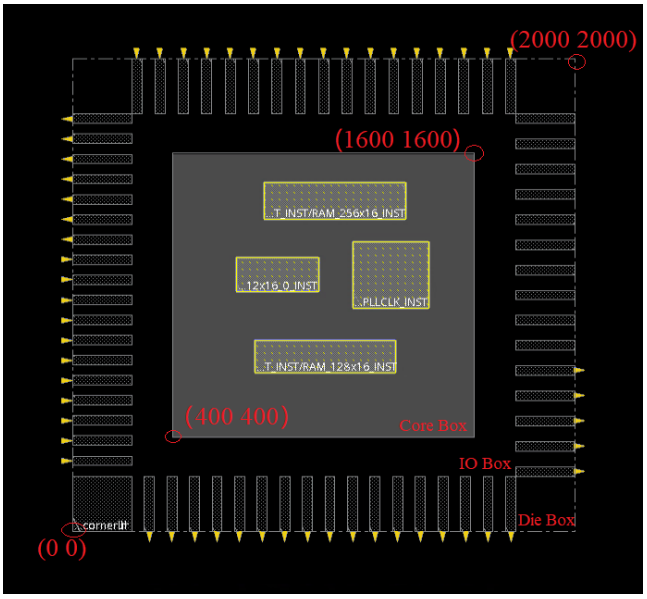
Note: The `create_floorplan` command checks if the specified IO box and core box is within limits of die box and displays an error message if they are outside the die box.

Parameters

`-box_size {die_box io_box core_box}`

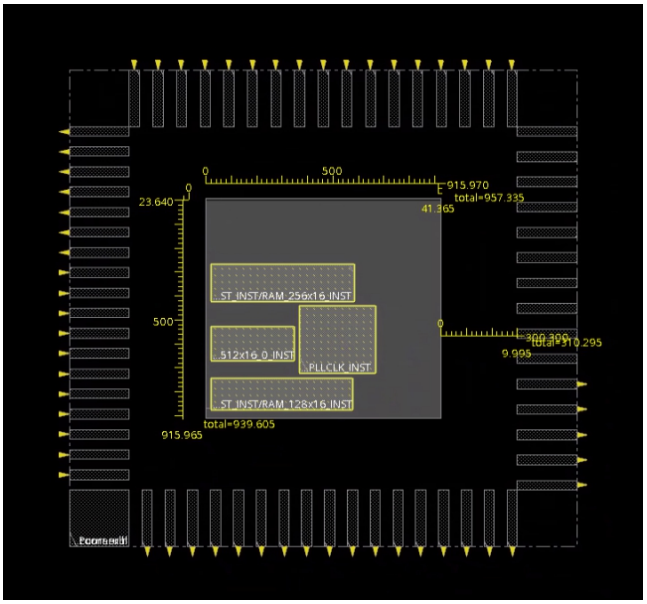
Specifies all three boxes of the floorplan, `die_box`, `io_box`, `core_box`. The following command creates floorplan by specifying a core box size of 1200 x 1200,io box size of 1600 x 1600 and die box size of 2000 x 2000.

```
create_floorplan -box_size {0 0 2000 2000 200 200 1800 1800 400 400 1600 1600}
```

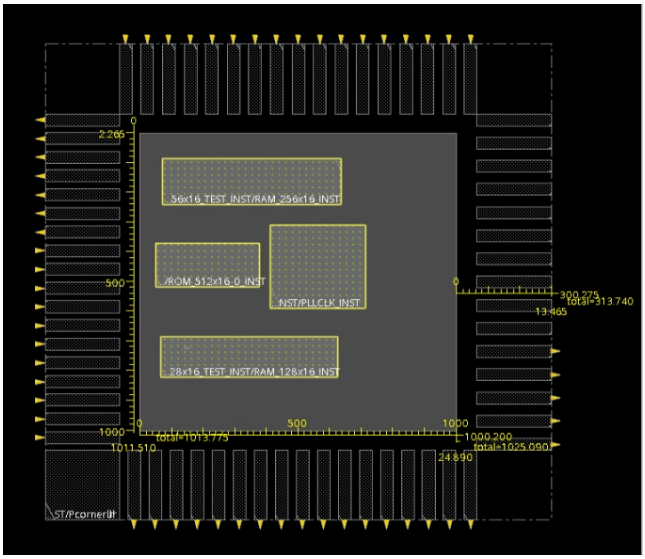


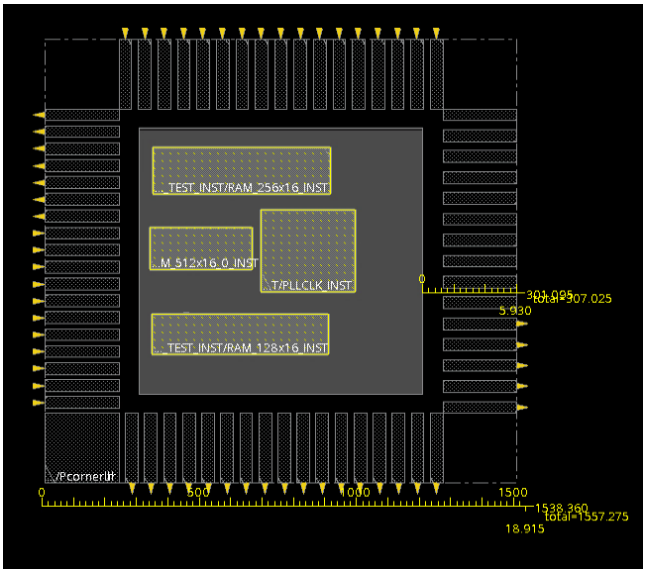
Data_type: list, required

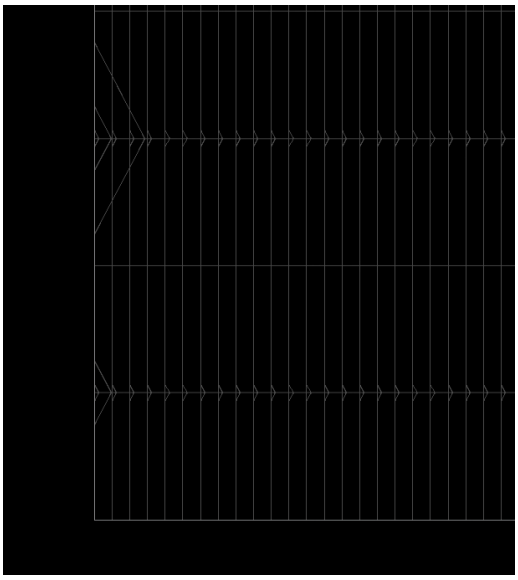
`-core_density_size {aspect_ratio [row_density [left bottom right top]]}`

	Specifies the chip's core dimensions as the ratio of the height divided by the width. If a value of 1.0 is used, a square chip is defined. A value of 2.0 will define a
	rectangular chip with height dimension that is twice the width dimension.
	• *aspect_ratio*: Specifies the aspect ratio value.
	• *row_density*: Specifies a row density value.
	`row_density = (std area + block/macro area) / core area`
	• *left*: Specifies the margin from the outside edge of the core box to the left.
	• *bottom*: Specifies the margin from the outside edge of the core box to the bottom.
	• *right*: Specifies the margin from the outside edge of the core box to the right.
	• *top*: Specifies the margin from the outside edge of the core box to the top.
	The following command creates floorplan by specifying the core dimensions aspect ratio of 1.0,row density of 0.5,and the spacing between core edge to each io box edge of 300.
	`create_floorplan -site tsm3site -core_density_size {1.0 0.5 300 300 300 300} -core_margins_by io`
	 
	
	Data_type: list, required

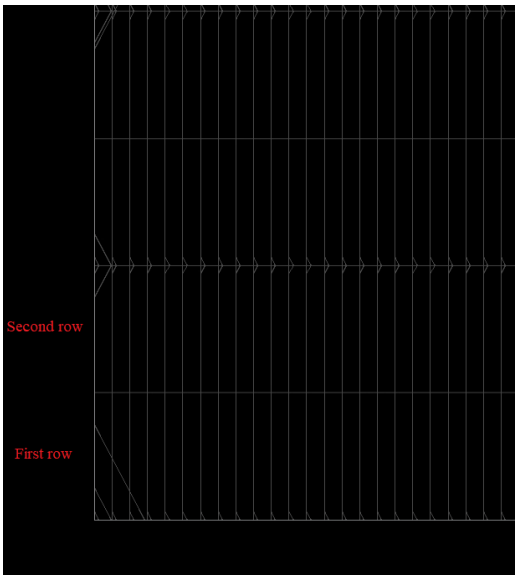
`-core_margins_by {io | die}`	
	Specifies whether the core margins are calculated using the core-to-IO boundary or the core-to-die boundary. *Default*: `io` *Data_type*: enum, optional
`-core_size {w h left bottom right top}`	

	Specifies the core size and the spacing, in micrometers, between the core edge, which is the margin between the outside edge of the core (head) box. • *w*: Specifies the core box's width value. • *h*: Specifies the core box's height value. • *left*: Specifies the margin from the outside edge of the core box to the left. • *bottom*: Specifies the margin from the outside edge of the core box to the bottom. • *right*: Specifies the margin from the outside edge of the core box to the right. • *top*: Specifies the margin from the outside edge of the core box to the top. The following command creates floorplan by specifying a core size of 1000 x 1000, and the spacing between core edge to each die edge of 300. `create_floorplan -site tsm3site -core_size{1000 1000 300 300 300 300} -core_margins_by die`  *Data_type*: list, required
-die_size {w h left bottom right top}	

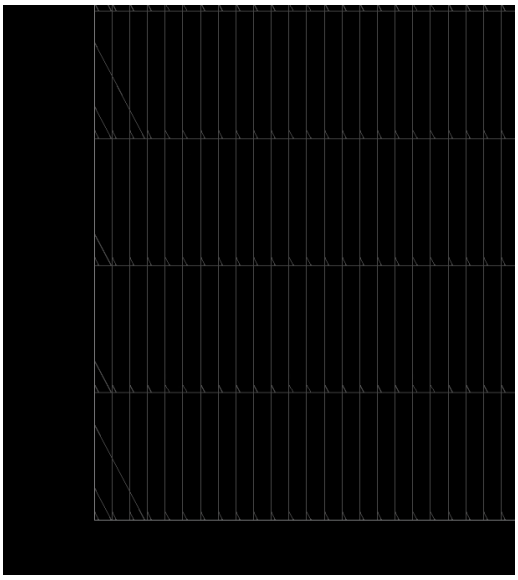
	Specifies the die size and the spacing, in micrometers, between the die edge, which is the margin between the outside edge of the die. • *w*: Specifies the die's width value. • *h*: Specifies the die's height value. • *left*: Specifies the margin from the outside edge of the core to the left of the I/O boundary. • *bottom*: Specifies the margin from the outside edge of the core to the bottom of the I/O boundary. • *right*: Specifies the margin from the outside edge of the core to the right of the I/O boundary. • *top*: Specifies the margin from the outside edge of the core to the top of the I/O boundary. The following command creates floorplan by specifying a die box size of 1500 x 1500,and the spacing between core edge to each die edge of 300. `creat_floorplan -die_size {1500 1500 300 300 300 300}`  *Data_type*: list, required
`-die_size_by_io_height {min | max}`	
	Specifies whether the maximum I/O height or the minimum I/O height should be used for die size calculation. *Default*: `min` *Data_type*: string, optional
`-flip {f | s | n}`	
	Specifies the orientation of the bottom row in the core area. • `f`: Specifies that the first row flips from the bottom up. • `n`: Specifies no row flipping. • `s`: Specifies that the second row flips from the bottom up. `create_floorplan -box_size {0 0 2000 2000 200 200 1800 1800 400 400 1600 1600} -flip f`



```
create_floorplan -box_size {0 0 2000 2000 200 200 1800 1800 400 400 1600 1600} -flip s
```



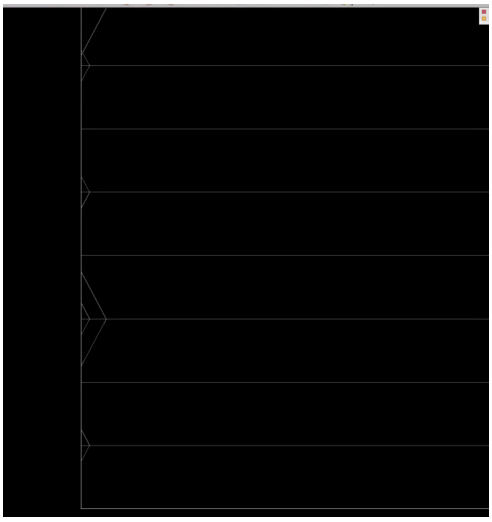
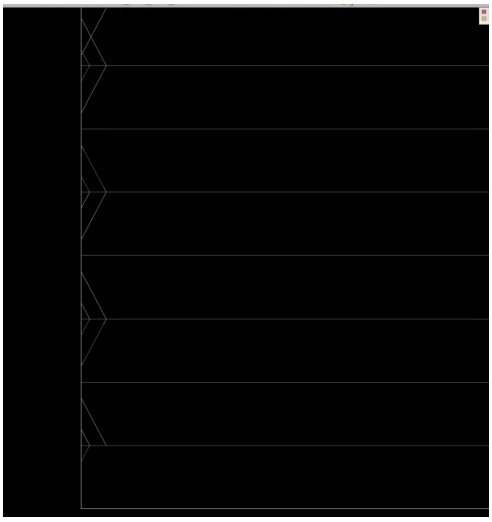

```
create_floorplan -box_size {0 0 2000 2000 200 200 1800 1800 400 400 1600 1600} -flip n
```



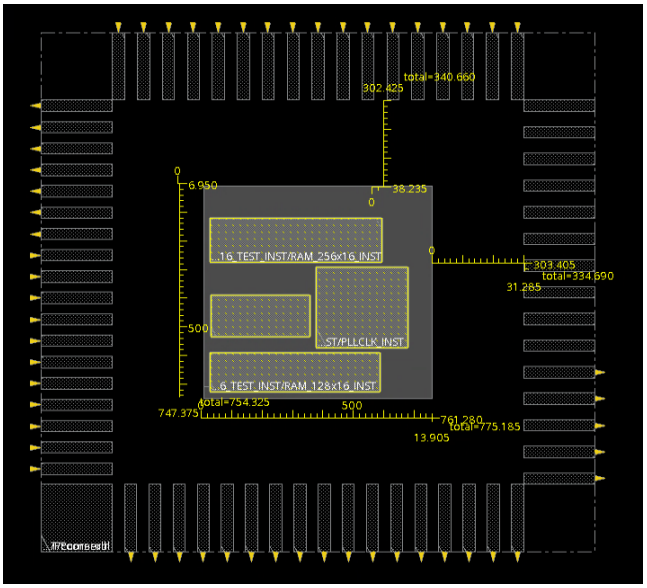
Default: `f`

Data_type: enum, optional

`-floorplan_origin {center | llcorner}`	
	Specifies whether the origin of the floorplan should be at the center or at the lower left corner. *Default*: `llcorner` (lower left corner) *Data_type:* enum, optional
`-help`	Outputs a brief description that includes the type and default information for each `create_floorplan` parameter. For a detailed description of the command and all of its parameters, use the man command: `man create_floorplan`
`-keep_shape_density` *`util`*	Specifies the target utilization after resize. *Data_type:* float, required
`-match_to_site`	Adjusts the width of the die area such that it is an integer multiple of the width of the IO site or core site. *Data_type*: bool, optional
`-no_resize`	Checks if there is a need to do floorplan if everything is the same. *Data_type*: bool, optional
`-no_snap_to_grid`	Specifies that the core box (or die box) boundary will not be snapped to the nearest metal pitch even if the if the specified die/ core box size is not an integer multiple of the smallest metal pitch. This prevents the input floorplan boxes from being changed. *Data_type*: bool, optional

`-overlap_same_site_row`	Specifies that same-site rows can overlap with one another. Single height rows must not overlap. Only double or multiple-height rows can overlap, and in sections that are multiple of single height any other overlap section size will cause the rows to be ignored. Example: `create_floorplan -core_size {1000 1000 300 300 300 300} -overlap_same_site_row` Before:  After:  *Data_type*: bool, optional

`-site name`	Specifies a core row site. *Data_type*: (site), optional
`-site_Only name`	Creates rows only for the specified site. *Data_type*: (site), optional
`-stdcell_density_size {aspect_ratio [std_cell_density [left bottom right top]]}`	

	Determines the core and module sizes by standard cell density.
	• *aspect_ratio*: Specifies the aspect ratio value.
	• *std_cell_density*: Specifies a standard cell density value.
	`std_cell_density = std cell area/(core area - block/macro area)`
	• *Left*: Specifies the margin from the outside edge of the core box to the left.
	• *Bottom*: Specifies the margin from the outside edge of the core box to the bottom.
	• *Right*: Specifies the margin from the outside edge of the core box to the right.
	• *Top*: Specifies the margin from the outside edge of the core box to the top.
	The following command creates floorplan by specifying the core dimensions aspect ratio of 1.0,standard cell density of 0.5,and the spacing between core edge to each io box edge of 300.
	`create_floorplan -stdcell_density_size {1.0 0.5 300 300 300 300}`
	 
	
	Data_type: list, required

Examples

- The following command creates floorplan by specifying a core size of 1000 x 1000, and the spacing between core edge to each die edge of 300.
  `create_floorplan -site tsm3site -core_size {1000 1000 300 300 300 300} -core_margins_by die`
- The following command creates floorplan by specifying the core dimensions aspect ratio of 1.0, core density of 0.5, and the spacing between core edge to each io box edge of 300.
  `create_floorplan -site tsm3site -core_density_size {1.0 0.5 300 300 300 300} -core_margins_by io`
- The following command creates a new floorplan containing all the sites in the original floorplan
  `create_floorplan -core_size {...}`
- The following command automatically shrinks or expands the block-level floorplan boundary, proportionally, in the X and Y directions, trying to meet the specified target core utilization 75.
  `create_floorplan -keep_shape_density 75`

Related Information

- `add_tracks`
- `check_floorplan`
- `floorplan` Category Attributes
- `read_floorplan`
- `init_core_rows`
- `write_floorplan`
- `delete_all_floorplan_objs`
- Floorplanning the Design chapter in the *Innovus Stylus Common UI User Guide*

create_group

```
create_group
[-help]
[-floating]
[-is_physical_hier]
-name group_name
[-type {fence region guide cluster}]
[-density value [-aspect_ratio value]]
[-rects {{x1 y1 x2 y2} ...} | -polygon {{x1 y1} {x2 y2} ...}]
```

Creates a new instance group, even outside the core boundary. Use this command after importing the design.

Note: Groups are used to create a new logical hierarchy unit, and normally all power domains associate with a group which always have a boundary constraint fence. The boundary constraint fence means a standard cell belonging to this group cannot be placed outside and only a standard cell belonging to this group can be placed inside. Cadence recommends that this physical-logical coherence should not be violated. The coherence requirement does not only apply for groups, but also pertains to all hinsts and groups as long as they have boundary constraint fence.
In other words, if fence A belongs to fence B in logical, then fence A must be placed inside fence B.

Parameters

`-aspect_ratio value`	Specifies the aspect ratio. If this is not specified, 1 is the default. *Data_type*: float, optional
`-density value`	Specifies a placement density percentage. Use the decimal format, where 0.75 is for 75% density. *Minimum*: `0` *Maximum*: `1` *Data_type*: float, optional
`-floating`	Creates a floating instance group. **Note:** It does not support rectilinear shapes. **Note:** You can use the `-floating` parameter for the `fence` or `region` type of instance groups only. *Data_type*: bool, optional
`-help`	Outputs a brief description that includes the type and default information for each `create_group` parameter. For a detailed description of the command and all of its parameters, use the man command: `man create_group`
`-is_physical_hier`	Specifies that the created group is a physical hierarchy. By default, the created group is not saved back to the netlist. *Data_type*: bool, optional

`-name group_name`	Specifies the name of the created group. A group is created to contain hierarchical instances, instances (leaf instances) or other groups.
	Note: In the nested partitions scenario, to add a module to be within a module, you must include the hinst name of the module in which you want the new module to be in, within the group name.
	For example,
	`create_group -name i_a/bc -type region`
	`update_group -name i_a/bc -add -hinsts i_a/b update_group -name i_a/bc -add -hinsts i_a/c`
	Data_type: string, required

`-polygon {{x1 y1} {x2 y2} ...}`	Specifies the point list of the polygon for *fence*, *region* and *guide*. *Data_type*: (point)+, optional

`-rects {{x1 y1 x2 y2} ...}`	Specifies the coordinates of the rectangle for *fence*, *region* and *guide*. • `x1`: Specifies the lower left x coordinate of the box. • `y1`: Specifies the lower left y coordinate of the box. • `x2`: Specifies the upper right x coordinate of the box. • `y2`: Specifies the upper right y coordinate of the box. **Note**: Since a fence should not cross multiple power domains, the `create_group` command should only be used to group objects in the same power domain (otherwise the design may become unplaceable). *Data_type*: (rect)+, optional
`-type {fence region guide cluster}`	Specifies the type of the group to create. You can specify: • `fence`: Specify to create a group of fences. • `region`: Specify to create a group of regions. • `guide`: Specify to create a group of guides. • `cluster`: Specify to create a group of clusters. *Data_type*: enum, optional

Examples

- The following command creates group adder1:
  ```
  create_group -name adder1 -type fence
  ```
- The following command creates group gusher_4 and its placement guide at the specified location:
  ```
  create_group -name gusher_4 -type guide -rects {2345.3 9962.1 4930.0 18244.8}
  ```
- The following command creates group adder2 with 80% of the area occupied by the instance area with an aspect ratio of 2:
  ```
  create_group -name adder2 -type cluster -density 0.8 -aspect_ratio 2
  ```

Related Information

- `update_group`
- `create_exclusive_groups`
- Floorplanning the Design chapter in the *Innovus Stylus Common UI User Guide*
  - Grouping Instances

create_interposer_route_block

```
create_interposer_route_block
[-help]
-box {x1 y1 x2 y2}
-cell cell_name
-dir dir_name
-inst master_name
```

Creates the master block. It transforms the routed domain to a master partition block that enrolls all the special routes.

Note: If the design does not have a routing pattern that can be treated as a master block, the functionality of the `create_interposer_route_block` command can be replaced by loading the LEF file of the master partition that has the same routing pattern using the `read_physical -add_lefs` command. After this, the `add_interposer_route_block` command can be used directly.

Parameters

`-box {x1 y1 x2 y2}`	Specifies the master block's bounding box. **Note**: The snapping grid of the created block honors the `the floorplan_snap_block_grid` attribute settings. *Data_type*: box, required
`-cell cell_name`	Specifies the master cell name. *Data_type*: string, required
`-dir dir_name`	Specifies the directory name that will contain the partition information and files. The tool creates the directory and saves the information and files for the top and master block partition automatically after the `create_interposer_route_block` command is used. *Data_type*: string, required
`-help`	Outputs a brief description that includes the type and default information for each `create_interposer_route_block` parameter. For a detailed description of the command and all of its parameters, use the man command: `man create_interposer_route_block`
`-inst master_name`	Specifies the master block name. **Note:** The default orientation of the created master block is R0. *Data_type*: string, required

Examples

- The following command transforms the routed domain to a master partition block.
  ```
  create_interposer_route_block -cell cell_name -inst master_block -box {llx lly urx ury} -dir PTN
  ```

Related Information

- `add_interposer_route_block`
- `read_physical`
- `floorplan_snap_block_grid` attribute
- "Repeating Routing Patterns in Interposer Designs" in the *Innovus Stylus Common UI User Guide*

create_io_row

```
create_io_row
[-help]
{-site site_name [{-side {N | W | S | E}
[-row_margin value]
[-begin_offset value]
[-end_offset value | -length len | -number_row_sites nr] } |
{-corner {BL | BR | TR | TL} [-x_offset value] [-y_offset value]}}]
[-orient {r0 | r90 | r180 | r270}]
[-name row_name]} | {-derive_by_cells} | {-derive_by_selection}
```

Creates new I/O rows and edits the existing I/O rows. You can use this command in the floorplanning stage.

Note: Before creating I/O rows for I/O row based pad placement, you need to set the I/O row flag using the `set_io_flow_flag` command.

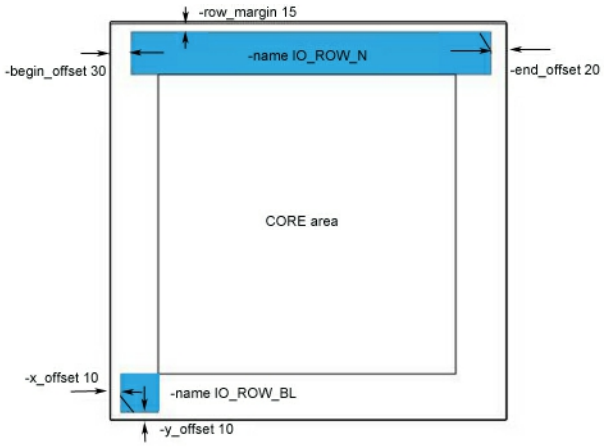
Parameters

`-begin_offset value`	Specifies the starting location of the I/O row from the die edge. • For a horizontal row, this value is the lower X value. • For a vertical row, this value is the lower Y value. *Default*: 0, starting from the die edge. *Data_type*: float, optional
`-corner {BL | BR | TR | TL}`	
	Specifies the location of the corner cell. This can be: • `BL`: Bottom Left • `BR`: Bottom Right • `TR`: Top Right • `TL`: Top Left *Default*: `BL` *Data_type*: string, optional
`-derive_by_cells`	Creates an I/O row for all the I/O cells in the design. *Data_type*: bool, required
`-derive_by_selection`	Creates an I/O row only for the selected I/O cells. **Note:** When you specify this parameter, some I/O rows may overlap depending on the I/O cell distribution. *Data_type*: bool, required
`-end_offset value`	Specifies the end location of the I/O row from the die edge. • For a horizontal row, this value is the higher X value. • For a vertical row, this value is the higher Y value. *Default*: 0, ending from the die edge. *Data_type*: float, optional
`-help`	Outputs a brief description that includes type and default information for each `create_io_row` parameter. For a detailed description of the command and all of its parameters, use the man command: `man create_io_row.`
`-length len`	Specifies the length of the row, in µmeters, starting from the die edge or `-begin_offset value`. *Data_type*: float, optional
`-name row_name`	Specifies the name of the I/O row to be created. *Data_type*: string, optional

`-number_row_sites` *nr*	Specifies the number of sites to define the length of the row. **Note:** You can either specify `-end_offset` or `-length` or `-number_row_sites` each time you run the `create_io_row` command. *Data_type*: int, optional
`-orient {r0 | r90 | r180 | r270}`	
	Specifies the orientation of the created row. If you do not specify this parameter, the default orientation for each side starting from the row on the south side is r0, rotate r90 for the next side, and so on... as follows: • r0 - South West • r90 - South East • r180 - North East • r270 - North West The default order of orientation is South, East, North, and West. *Data_type*: string, optional
`-row_margin` *value*	Specifies the distance, in µmeters, from the row edge to the die edge. The row edge is `top' for a North side row, `bottom' for a South side row, `left' for a West side row, and `right' for an East side row. *Data_type*: float, optional
`-side {N | W | S | E}`	Specifies the side on which the row is to be created. If you do not specify the side, by default, one row is created on all sides (South, East, North, and West). *Data_type*: string, optional
`-site` *site_name*	Specifies the name of the site defined in the LEF library. *Data_type*: string, required
`-x_offset` *value*	Specifies the distance, in µmeters, from the left or right edge of the corner row to the die edge. *Data_type*: float, optional
`-y_offset` *value*	Specifies the distance, in µmeters, from the top or bottom edge of the corner row to the die edge. *Data_type*: float, optional

Example

- The following command creates I/O rows based on the existing pad placement.
  ```
  create_io_row -derive_by_cells
  ```
- The following command creates I/O row at north side with offset 30 from left side, offset 20 from right side and row margin 15 from top side.
  ```
  create_io_row -name IO_ROW_N -begin_offset 30 -end_offset 20  -row_margin 15 -side N -site IO -orientation r180
  ```
- The following command creates I/O row at bottom left corner to place corner IO with offset 10 in x and y direction.
  ```
  create_io_row -name IO_ROW_BL  -x_offset  10  -y_offset  10  -corner BL -site corner -orientation r0
  ```



Related Information

- `set_io_flow_flag`
- `add_area_io_fillers`
- `get_io_flow_flag`
- Floorplan Menu chapter in the *Innovus Stylus Common UI Menu Reference*
  - Create I/O Row
- Floorplanning the Design chapter in the *Innovus Stylus Common UI User Guide*
  - Performing I/O Row Based Pad Placement
- Design Implementation Flow chapter in the *Innovus Stylus Common UI User Guide*
  - Creating an I/O Assignment Flow

create_logic_hierarchy

```
create_logic_hierarchy
[-help]
[-commit]
-hinst_name string
-module_name string
[-objs string]
[-selected]
```

Creates a new logical hierarchy for the specified hierarchical instances. The `create_logic_hierarchy` command changes the netlist, CPF, SDC, Scan DEF, and floorplan constraints for the newly created hierarchy.

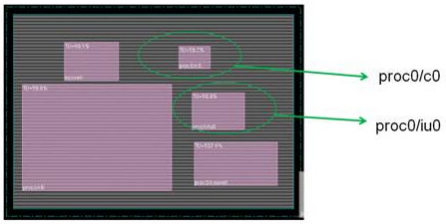
Parameters

`-help`	Outputs a brief description that includes the type and default information for each `create_logic_hierarchy` parameter. For detailed description of the command and all its parameters, use the man command: `man create_logic_hierarchy`.
`-module_name string`	Specifies the new cell name. *Data_type*: string, optional
`-commit`	Specifies the commit group result. *Data_type*: bool, optional
`-hinst_name string`	Specifies the new hier inst name. *Data_type*: string, required
`-objs string`	Specifies the list of hinst and/or instance groups. *Data_type*: string, optional
`-selected`	Specifies that grouping is done for the current selected hinsts or instant groups. *Data_type*: bool, optional

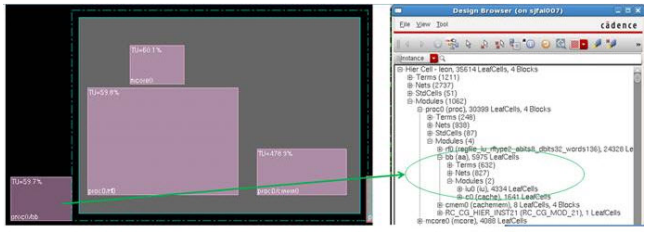
Examples

- After using the following command, you can create a new Hinst name `proc0/bb` whose cell name is `aa` and add module `proc0/c0` and `proc0/iu0` to Hinst `proc0/bb`.

  ```
  create_logic_hierarchy -commit -module_name aa -hinst_name proc0/bb -objs {proc0/c0 proc0/iu0}
  ```

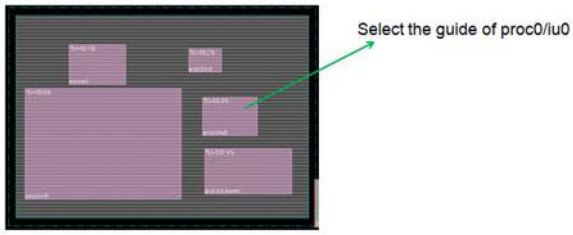
  

  The module `proc0/c0` and `proc0/iu0` will be moved to group `proc0/bb`.

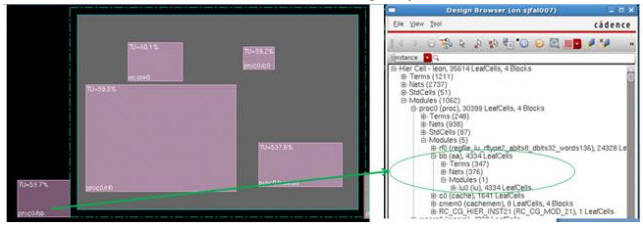
  

- After using the following command, you can create a new Hinst name `proc0/bb` whose cell name is `aa` and add selected module (it is `proc0/iu0` in this case) to Hinst `proc0/bb`.

```
create_logic_hierarchy -commit -module_name aa -hinst_name proc0/bb -selected
```



The module `proc0/iu0` will be moved to group `proc0/bb`.



Note: It will not create group result without `-commit` option of `create_logic_hierarchy` command. Without the `-commit` option, the netlist of the design is not changed.

Related Information

- Floorplanning the Design chapter in the *Innovus Stylus Common UI User Guide*
  - Adding Logical Hierarchy Without Creating Additional Hierarchy
  - Logical Hierarchy Manipulation

create_pg_model_for_macro_place

```
create_pg_model_for_macro_place
[-help]
{-file string}
```

Enables you to create pg models for the concurrent macro placement. This command can save pg modeling to a Tcl file from the existing floorplan. You can use this command after the existing floorplan has been power planned.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `create_pg_model_for_macro_place` parameter. For a detailed description of the command and all of its parameters, use the man command: `man create_pg_model_for_macro_place`
`-file string`	Specifies the name of the file to which PG model details are written. *Data_type*: string, required

Example

After restoring the initial DB with power mesh, the following command can be used to create pg models for macro placement;

```
create_pg_model_for_macro_place -file golden_mimic_power_mesh.tcl
```

Related Information

- `check_macro_place_constraint`
- `get_macro_place_constraint`
- `reset_macro_place_constraint`
- `set_macro_place_constraint`
- `write_macro_place_constraint`
- Floorplanning the Design chapter in the *Innovus Stylus Common UI User Guide*
- Using the Mixed Placer chapter in the *Innovus Stylus Common UI User Guide*

create_pg_pin

```
create_pg_pin
[-help]
{-on_die {-selected | -net net_name}
[-width float]
[-length float]} |
{-name [-net net_name]
[-geometry layername x1 y1 x2 y2]}
[-dir {undefined | input | output}]
```

Creates a power/ground pin as per the specified coordinates of the physical shape. You can use this command to create a power/ground pin without having to first create a power stripe with the `add_stripes` command.

- If the `-geom` parameter is not specified, only the logical pin is created.
- If the `-geom` parameter is specified, the physical pin is also created.
- If a logical power/ground pin exists before this command is run, the net name will be ignored if it is specified.
- If a logical power/ground pin does not exist before this command is run:
  - If a net has been specified with the -net parameter, the power/ground pin will be created and attached to the net specified.
  - If a net name has not been specified, the net name is assumed to be the same as the pin name. In this case, the power/ground pin will be created and attached to that nets.

Parameters

`-dir {undefined | input | output}`	
	Specifies the direction of the power/ground pin. You can specify one of the following: • `undefined`: Specifies that the direction of the pin is not defined. • `input`: Specifies that the direction of the pin is input. • `output`: Specifies that the direction of the pin is output. *Data_type:* enum, optional
`-geometry layername x1 y1 x2 y2`	
	Specifies the geometry of the physical pin. • *`layername`*: Specifies the layer on which the power/ground pin will be created. The *`layeid`* can range from 1 to n, where, the maximum value for n is the top most routing layer. **Note:** Layers can also be specified using layer names (for example, Metal1, Metal2, Metal3) or the LEF layer names. • *`x1`*: Specifies the lower-left x coordinate, in microns, of the power/ground pin. • *`y1`*: Specifies the lower-left y coordinate, in microns, of the power/ground pin. • *`x2`*: Specifies the upper-right x coordinate, in microns, of the power/ground pin. • *`y2`*: Specifies the upper-right y coordinate, in microns, of the power/ground pin. *Data_type:* string, optional
`-help`	Outputs a brief description that includes the type and default information for each *`create_pg_pin`* parameter. For a detailed description of the command and all of its parameters, use the man command: `man create_pg_pin`
`-length float`	Specifies the length of the power/ground pin. *Default*: Same as the width *Data_type:* float, optional
`-net net_name`	Specifies the name of the net to which the power/ground pin will be attached. *Data_type:* string, optional
`-on_die`	Creates power/ground pin on the die. *Data_type:* bool, optional
`-name`	Specifies the name of the power/ground pin. *Data_type:* string, optional
`-selected`	Creates power/ground pin on the die, for all selected stripes. *Data_type:* bool, optional
`-width float`	Specifies the width of the power/ground pin. *Default*: width of the stripe. *Data_type:* float, optional

Examples

- This example is for the case where a logical pin name has not been defined prior to running the command and a net name has been specified.
  The following command creates a power/ground pin named pgpin_A on layer Metal2 and attaches the pin to the net net_A. The pin coordinates are 7 (x1), 3 (y1), 17 (x2), and 23 (y2).
  ```
  create_pg_pin -geometry Metal2 7 3 17 23 -net net_A pgpin_A
  ```
- This example is for the case where a logical pin name has not been defined prior to running the command and a net name has not been specified.
  The following command creates a power/ground pin named pgpin_B on layer Metal3 and attaches the pin to the net pgpin_B. The pin coordinates are 7 (x1), 3 (y1), 17 (x2), and 23 (y2). As the net name was not specified, the net name is assumed to be the same as the pin name.

```
create_pg_pin -geometry Metal3 7 3 17 23 pgpin_B
```

- This example is for the case where a logical pin name has already been defined prior to running the command.
  The following command creates a power/ground pin named pgpin_C on layer Metal5. The pin coordinates are 7 (x1), 3 (y1), 17 (x2), and 23 (y2). In this case, the logical pin name was already defined prior to running the command, and so a net name is not required. Even if a net name were specified, it would have been ignored.
  ```
  create_pg_pin -geometry Metal5 7 3 17 23 pgpin_C
  ```
- This example is for the case where the net name is not specified and the -geom parameter has not been specified.
  The following command creates a logical pin names pgpin_D. The net name is assumed to be the same as the pin name and no physical pin is created.
  ```
  create_pg_pin pgpin_D
  ```
- This example is for the case where the pin is created on the die attached to net vdd with a width of 2 and a length of 3.
  ```
  create_pg_pin -on_die -net vdd -width 2 -length 3
  ```

Related Information

- add_stripes
- delete_pg_pins
- Floorplanning the Design chapter in the *Innovus Stylus Common UI User Guide*
- Floorplan Menu chapter in the *Innovus Stylus Common UI Menu Reference*
  - Create P/G Pin

create_physical_pin

```
create_physical_pin
[-help]
[-allow_outside_boundary]
-name pin_name
[-net net_name]
[-same_port]
{{-layer layer_name {-rect {x1 y1 x2 y2} | -polygon {x1 y1 x2 y2 ... xn yn}} } }
```

Adds a new physical pin-shape to a top-level port (DEF PIN). You can use this command to add more PG (power/ground) pin-shapes to an existing PG port. It will also create a new PG port if there is not already a PG port.

Note: It can add shapes to top-level signal ports, but this is not recommended. Multiple shapes for a signal port may cause flow problems (the router will not connect the different pin shapes, etc.) and should only be done with expert knowledge of the flow implications. It will not create a new signal port (You must use ECO commands to add a new signal port).

Parameters

`-allow_outside_boundary`	Do not check whether the specified rectangle is inside the design boundary. *Data_type:* bool, optional
`-help`	Outputs a brief description that includes the type and default information for each `create_physical_pin` parameter. For a detailed description of the command and all of its parameters, use the man command: `man` create_physical_pin
`-layer layer_name`	Specifies the LEF (or OA) layer name, or the routing layer index value. For example, 1 for first routing layer, 2 for second routing layer. *Data_type:* string, optional
`-name pin_name`	Specifies the name of the top-level port (DEF PIN name) to add shapes to. If it does not exist, then `-net net_name` is used to find a top-level port instead. If `net_name` does not already have a top-level port, and it is a PG net, then a new port named `pin_name` will be created for `net_name`. *Data_type:* string, required
`-net net_name`	Specifies the name of the net. This is ignored if a top-level port named *pin_name* already exists. *Data_type:* string, optional
`-polygon {x1 y1 x2 y2 ... xn yn}`	Specifies the coordinates of each vertex of the shape. *Data_type:* (point)+, optional
`-rect {x1 y1 x2 y2}`	Specifies the coordinates of the rectangular shape. • *x1* Specifies the lower-left x coordinate, in microns, of the physical pin. • `y1`: Specifies the lower-left y coordinate, in microns, of the physical pin. • `x2`: Specifies the upper-right x coordinate, in microns, of the physical pin. • `y2`: Specifies the upper-right y coordinate, in microns, of the physical pin. *Data_type:* rect, optional
`-same_port`	Specifies that the physical pin shapes will be added to the first DEF PORT. If this parameter is not specified, each pin shape is created on a separate DEF port. Normally each PG shape should be on a separate port so each one is connected separately to the PG mesh at the next level up. *Data_type:* bool, optional

Examples

- The following command adds a physical pin-shape to the top-level term named VSS (e.g. DEF PIN named VSS).
  ```
  create_physical_pin -name VSS -layer Metal2 -rect {7 3 17 23}
  ```

Related Information

- Floorplanning the Design chapter in the *Innovus Stylus Common UI User Guide*

create_place_blockage

```
create_place_blockage
[-help]
{ { [ -polygon {{x1 y1} {x2 y2}...} | -rects { { lx1 ly1 ux1 uy1 } { lx2 ly2 ux2 uy2 } ... }
    | -all_partitions | -all_macros ]
    [-inst  inst_name  | -hinst  hinst_name ] }
    [-name  place_blockage_name  | -use_prefix ]
    [{-type {hard soft partial macro_only}} [-density  value ] [-exclude_flops ]]
    [-no_cut_by_core ] [-snap_to_site ] }
[{-inst  inst_name  | -hinst  hinst_name  | -all_partitions | -all_macros }
 [-cover ]
[-inner_ring_by_side { x1 y1 x2 y2 } | -inner_ring_by_edge  { edge1 edge2 edge3 ... } ]
[-outer_ring_by_side { x1 y1 x2 y2 } | -outer_ring_by_edge  { edge1 edge2 edge3 ... } ] ]
```

Creates cell placement blockages that can be placed. A placement blockage is a floorplan object used to block standard cell placements. Use this command after importing the design.

Parameters

`-all_macros`	For each macro creates a placement blockage covering that macro (.is_macro attribute on inst db objects). *Data_type*: bool, optional
`-all_partitions`	For each partition creates a placement blockage covering that partition. *Data_type*: bool, optional
`-cover`	By default the parameters (`-outer_ring_by_edge` and `-outer_ring_by_side`) used to create placement blockages around the outside of the object, create toroidal placement blockages. The `-cover` parameter combined with these parameters creates a solid placement blockage, otherwise the `-cover` parameter does nothing. *Data_type*: bool, optional
`-density value`	Specifies the maximum density allowed for partial or soft blockages. The value is in steps of 5 between 5 and 100, with units of percent. For example, a partial placement value of 75 means that up to 75 percent placement density is allowed in the area. **Note**: The `-density` and `-exclude_flops` parameters can only be used in combination with the `-type partial` or `-type soft` parameter. *Minimum*: 5 *Maximum*: 100 *Data_type*: inst, optional
`-exclude_flops`	Specifies that flops and latches will not be placed in the blockage during global placement. **Note**: The `-density` and `-exclude_flops` parameters can only be used in combination with the `-type partial` or `-type soft` parameter. *Data_type*: bool, optional
`-help`	Outputs a brief description that includes the type and default information for each `create_place_blockage` parameter. For a detailed description of the command and all of its parameters, use the man command: `man create_place_blockage`
`-hinst hinst_name`	Specifies the name of the hierarchical instance on which the placement blockage is to be applied. An `hinst` db object works too. *Data_type*: (hinst), optional
`-inner_ring_by_edge {edge1 edge2 edge3 ...}`	

	Creates a hollow ring of placement blockage around the inside of the object(s) specified. *edgeN* is the width in microns of the blockage created inside the Nth edge. *edge1* is the vertical edge starting at the left corner of the leftmost edge with the lowest y coordinate as placed, and proceeding clockwise through the other edges. **Note** : • The `-inst` , `-hinst` , `-all_macros`, or `-all_partitions` parameters must be specified with this parameter. • There must be an *edgeN* specified for every edge of the object(s). • Enclose the *edgeN*s in double-quotes, curly braces, or [list ...]. *Data_type*: string, optional
`-inner_ring_by_side {x1 y1 x2 y2 }`	
	Creates a hollow ring of placement blockage around the inside of the object(s) specified. *bottom* is the width in microns of the blockage created on bottom edges, etc. **Note**: • The `-inst`, `-hinst`, `-all_macros`, or `-all_partitions` parameters must be specified with this parameter. • `The -inner_ring_by_side` parameter can be used with rectilinear objects. • Enclose the *bottom...left* values in double-quotes, curly braces, or [list ...]
`-inst inst_name`	Specifies the name of the instance on which the placement blockage is to be applied. An `inst` db object works too. *Data_type*: (inst), optional
`-name place_blockage_name`	
	Specifies the name of the placement blockage. **Note:** Placement blockage names need not be unique. **Note:** If neither `-name` nor `-use_prefix` parameters are specified, then the created place blockage is named `defScreenName`. *Data_type*: string, optional
`-no_cut_by_core`	Specifies that placement blockages should not be truncated at the core boundary. Placement blockage will always be truncated at the die boundary. *Data_type*: bool, optional
`-outer_ring_by_edge { edge1 edge2 edge3 ...}`	
	Creates a hollow ring of placement blockage around the outside of the object(s) specified. *edgeN* is the width in microns of the placement blockage along the Nth edge. *edge1* is the vertical edge starting at the left corner of the leftmost edge with the lowest y coordinate as placed, and proceeding clockwise through the other edges. **Note**: • The `-inst`, `-hinst` , `-all_macros`, or `-all_partitions` parameters must be specified with this parameter. • There must be an *edgeN* specified for every edge of the object(s). • By default, a toroid of blockage is created around the specified object(s). Use the `-cover` parameter to create a solid blockage over the objects with the specified additional halo of blockage. • Enclose the *edgeN*s in double-quotes, curly braces, or [list ...]. *Data_type*: string, optional
`-outer_ring_by_side {x1 y1 x2 y2 }`	

	Creates a hollow ring of placement blockage around the outside of the object(s) specified. `bottom` is the width in microns of the placement blockage along the bottom edge(s) of object(s) specified, etc. **Note**: • The `-inst`, `-hinst`, or `-all_partitions` parameters must be specified with this parameter. • The `-outer_ring_by_side` parameter can be used with rectilinear objects. • By default, a toroid of blockage is created around the specified object(s). Use the -cover parameter to create a solid blockage over the objects with the specified additional halo of blockage. • Enclose the *bottom...left* in double-quotes, curly braces, or [list ...].
`-polygon {{ x1 y1 } { x2 y2 } ...}`	
	Specifies the polygon vertices of the blockage, in microns. *Data_type*: (point)+, optional
`-rects { { lx1 ly1 ux1 uy1 } { lx2 ly2 ux2 uy2 } ... }`	
	Specifies the box vertices of the blockage, in microns. *Data_type*: (rect)+, optional
`-snap_to_site`	Snaps the newly created placement blockage to a row site. *Data_type*: bool, optional
`-type { hard | soft | partial | macro_only}`	
	Specifies the type of blockage to be created. • `hard`: Neither standard cell instances nor macros may be placed in the blockage. This is the default. • `soft`: Only buffers, inverters, isolation cells, clock gates, tie cells and level shifters are allowed to be placed inside the soft blockages. All other cells are placed outside the soft placement blockage. You can optionally specify cells and instances that are allowed or forbidden to be placed inside soft blockages by using the `set_selective_blockage_gate` or `reset_selective_blockage_gate` commands. • `partial`: In conjunction with the `-density` parameter alone or with the `-density` and `-exclude_flops` parameters together, creates a placement blockage that has a maximum density as specified (and sometimes excludes flops and latches too). • `macro_only`: Enables the `proto_design` command to keep macros out of the placement blockage; however, it enables standard cells to be placed inside the box as if no blockage is present. Macro-Only blockage is represented as "0" percent partial blockage in floorplan file, DEF file, and in Innovus DB. *Default*: `hard` **Note**: Alternatively, you can create different types of placement blockages using the *Set Placement Blockage Options* form or after creating the placement blockage, you can double-click on the object in the design display area to open the *Attribute Editor* and select from four blockage types. *Data_type*: enum, optional
`-use_prefix`	Specifies that the placement blockages should be named with a prefix defined by `floorplan_default_blockage_name_prefix` attribute. Names are of the form `prefix_<int>` *Data_type*: bool, optional

Examples

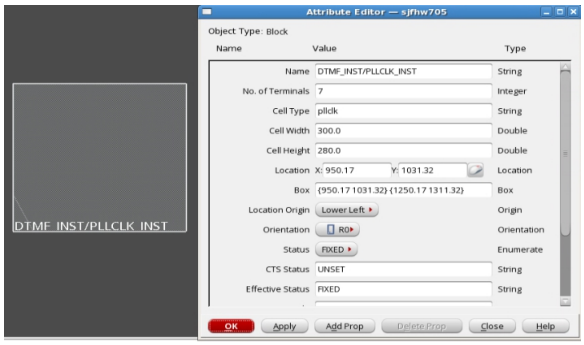
- The following command creates a soft blockage, `softBlockage2` in the design area:

```
create_place_blockage -type soft -rects { { 3442.3600 3739.2000 3511.6500 5716.6300 } } -name softBlockage2
```

- The following command creates a hard blockage covers the block instance of DTMF_INST/PLLCLK_INST:

```
create_place_blockage -type hard -inst DTMF_INST/PLLCLK_INST -cover
```

Before:

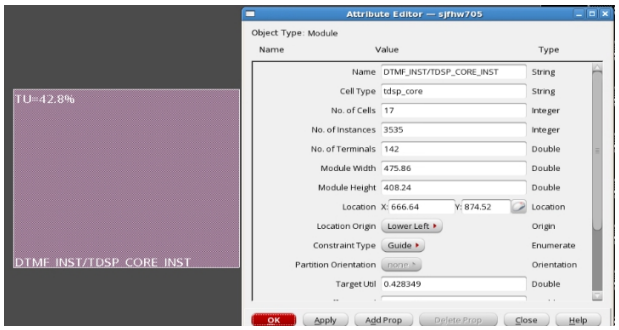


After:

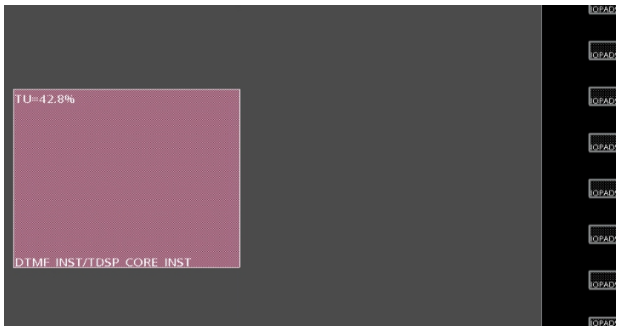
- The following command creates a hard blockage for hierarchical instance (Module) of DTMF_INST/TDSP_CORE_INST:

```
create_place_blockage -type hard -hinst DTMF_INST/TDSP_CORE_INST -cover
```

Before:



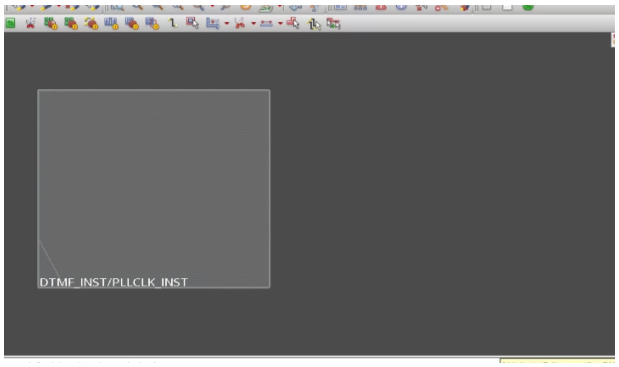
After:



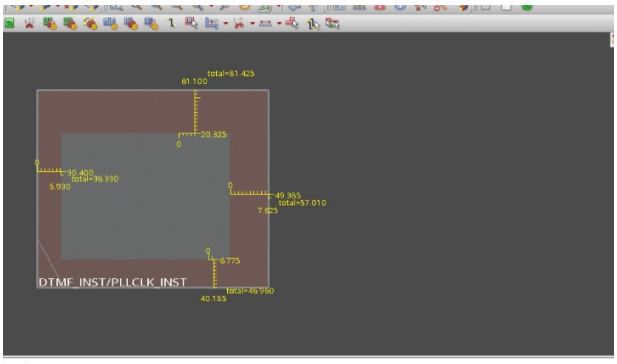
- The following command creates a hollow ring of placement blockage inside of the object:

```
create_place_blockage -type hard -inst DTMF_INST/PLLCLK_INST --inner_ring_by_edge {30 40 50 60}
```

Before:



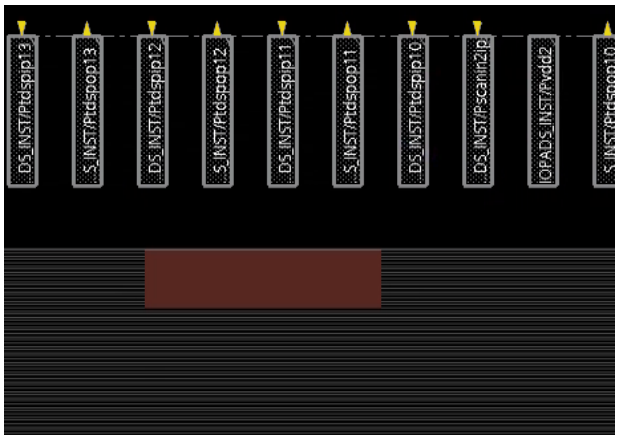
After:

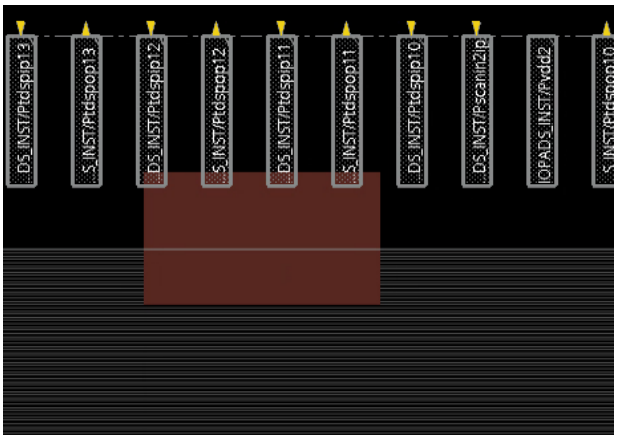


- The following command shows the functionality of the option `noCutByCore`:

```
create_place_blockage -rects {{952 1690 1291 1899}}
```

```
create_place_blockage -rects {{952 1690 1291 1899}} -no_cut_by_core
```





Related Information

- `delete_obj`
- `floorplan` Category Attributes
- Floorplanning the Design chapter in the *Innovus Stylus Common UI User Guide*

create_place_halo

```
   create_place_halo
[-help]
-halo_deltas left bottom right top
[-snap_to_site]
[-orient orientation]
[-insts name | -all_macros | -all_black_boxes | -all_commit_partitions | -all_blocks | -all_io_pads | -cell name]
```

Adds a halo to a block. A halo is an area that prevents the placement of standard cells within the specified halo distance from the edges of a hard macro, black box, or committed partition in order to reduce congestion. A block halo value is specified based on the current block orientation. Use this command during or after floorplanning. You can use the `delete_place_halo` command to delete halos.

Parameters

`-all_black_boxes`	Add halo around all black boxes. *Data_type*: bool, optional
`-all_blocks`	Add halo around all hard macros, black boxes, and committed partitions. *Data_type*: bool, optional
`-all_commit_partitions`	Add halo around all committed partitions. *Data_type*: bool, optional
`-all_io_pads`	Add halo around all IO pads. *Data_type*: bool, optional
`-all_macros`	Add halo around all hard macros. *Data_type*: bool, optional
`-cell name`	Add halo around all instances of cell `name`. *Data_type*: string, optional
`-halo_deltas left bottom right top`	
	Specifies the halo width on 4 sides. These values are floats, units are microns. Halo deltas are specified in relation to a block in R0 orientation (`left` is applied to the left edge of block oriented R0). For non-R0, the halo deltas are rotated/flipped according to the blocks's orientation (for block oriented MY, the `left` halo delta specifies the width of the halo along the *right* side of the inst in the floorplan). See `-orient` • `left`: the width of the halo along the left edge of insts having halos added. • `bottom`: the width of the halo along the bottom edge of objects having halos added. • `right`: the width of the halo along the right edge of objects having halos added. • `top`: the width of the halo along the top edge of objects having halos added. *Data_type*: rect, required
`-help`	Outputs a brief description that includes the type and default information for each `create_place_halo` parameter. For a detailed description of the command and all of its parameters, use the man command: `man create_place_halo`
`-insts name`	Add halo around named insts. You can use wildcards (* or ?) for specifying instance names. You cannot specify a list of names. You cannot use objects instead of names. *Data_type*: insts, optional
`-orient orientation`	Pre-rotate/flip the halo deltas using `orientation` before applying them. `orientation` can be `R0, R90, R180, R270, MX, MX90, MY, or MY90` (lower case values work too). *Data_type*: string, optional
`-snap_to_site`	Snaps the halo to site boundaries. *Data_type*: bool, optional

Example

The following command adds a halo around block `xy_inst`:

```
create_place_halo -halo_deltas 10 20 30 40 -insts xy_inst
```

If `xy_inst` is R0, the halo width is 10 along the left edge and 30 along the right edge. If the block is MY, the halo width is 30 along the *left* edge in the floorplan, and 10 along the *right* edge in the floorplan.

The following command adds a halo around all insts with cells of type `myDRAMCell`:

```
create_place_halo -halo_deltas 10 20 30 40 -cell myDRAMCell -orient MY
```

If the inst is oriented R0, then the *left* edge width of the halo is 30 in the floorplan, and the *right* edge width of the halo is 10.

Related Information

- `delete_place_halo`
- Floorplanning the Design chapter in the *Innovus Stylus Common UI User Guide*
- Floorplan Menu chapter in the *Innovus Stylus Common UI Menu Reference*
  - Edit Halo

create_relative_floorplan

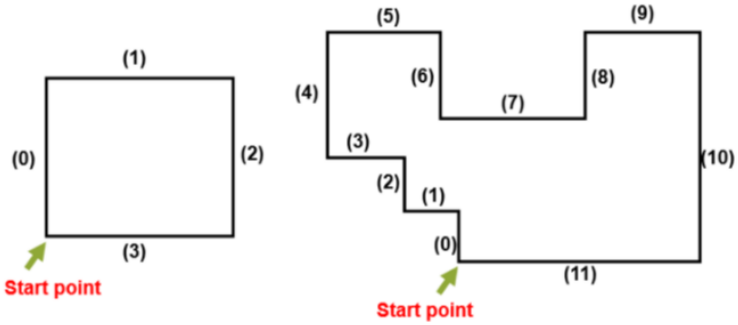
```
create_relative_floorplan
[-help]
{{{-place obj_name_list | -reshape obj_name_list} [-ref ref_obj_name_list]}}
[-place obj_name_list {-ref_type {{object core_boundary die_boundary}...}
 -horizontal_edge_separate {{ref_edge_horizontal y_offset obj_edge_horizontal}...}
 -vertical_edge_separate {{ref_edge_vertical x_offset obj_edge_vertical}...}}
[[-bbox {{none target reference both}...}]
 [-orient {{r0 r90 r180 r270 mx my mx90 my90}...}]]]
[-reshape obj_name_list { -dimension {{width height}...} }
[ [-fixed_edges {{edge1 edge2}...}] [-density {value}] [-offset {value}]]]
```

The `create_relative_floorplan` command captures and defines the placement relationship of floorplan objects independently from the actual coordinates in a floorplan, and resizes modules or blackboxes based on other floorplan objects, even outside the core boundary. You can use this command after importing the design.

Note: The `create_relative_floorplan` command accepts an array of objects. You can capture and define the placement relationship of an array of floorplan objects using just one command.

Note: You can use `write_floorplan_script -sections relative_fplan` to save relative floorplan info.

Note: The edges are numbered as 0,1,2,3... from the low left corner in the clockwise direction. The left edge closest to the origin that has the smallest x,y coordinates is taken as the first zero edge. An even edge number donates a vertical edge and an odd edge number donates a horizontal edge. The following figure shows how to count the edges for a rectangle and rectilinear object.



Parameters

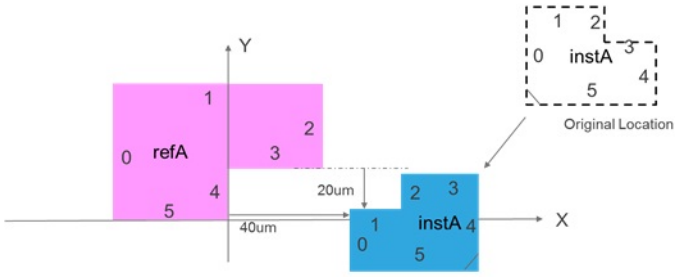
`-bbox {none target reference both}`	
	Specifies whether to use a bounding box instead of the rectilinear shape of target or reference object. *Default*: `none` *Data_type:* (enum)+, optional
`-density value`	Specifies the target utilization (TU). By default, reshaping the object will keep the current TU. *Data_type:* string, optional
`-dimension {width height}`	Specifies the width or height of the referenced object as the reference dimension. *Default*: `width` *Data_type:* (enum)+, optional
`-fixed_edges {edge1 edge2}`	Specifies that the reshape object will be stretched while fixing two perpendicular edges. *Default*: `{-1 0}` *Data_type:* string, optional
`-horizontal_edge_separate {ref_edge_horizontal y_offset obj_edge_horizontal}`	
	Specifies the vertical spacing between the horizontal edge of the target and reference objects. *Data_type:* string, optional
`-help`	Outputs a brief description that includes the type and default information for each `create_relative_floorplan` parameter. For a detailed description of the command and all of its parameters, use the man command: `man create_relative_floorplan`
`-offset value`	Specifies the offset value from the reference object in the specified dimension. *Data_type:* string, optional
`-orient {r0 r90 r180 r270 mx my mx90 my90}`	
	Specifies the orientation of the target object. *Default*: `r0` *Data_type:* (enum)+, optional
`-place obj_name_list`	Specifies the target object to place. It can be hInst, inst, group, power_domain, pin_guide, blockages, or port. *Data_type:* string, optional
`-ref ref_obj_name_list`	Specifies the reference object name. It can be hInst, inst, group, power_domain, pin_guide, blockages, port, or pin. It cannot be used with `-ref_type core_boundary` or `-ref_type die_boundary`. *Data_type:* string, optional
`-ref_type {object core_boundary die_boundary}`	
	Specifies the type of reference. *Default:* `object` *Data_type:* (enum)+, optional
`-reshape obj_name_list`	Specifies the target object to reshape. It can be hInst, group, power_domain, pin_guide, or blockages. *Data_type:* string, optional
`-vertical_edge_separate {ref_edge_vertical x_offset obj_edge_vertical}`	
	Specifies the horizontal spacing between the vertical edge of the target and reference objects. *Data_type:* string, optional

Examples

Placing an Instance According to a Rectilinear Object

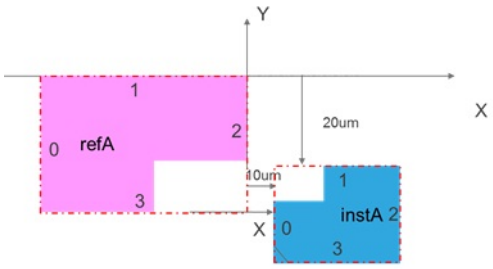
- The following command places an object with respect to a rectilinear object:

```
create_relative_floorplan -place instA -ref_type object -ref refA -orient MY \
        -horizontal_edge_separate {3 -20 1} -vertical_edge_separate {4 40 0}
```



- The following command places an object with respect to the bounding box of a rectilinear object:

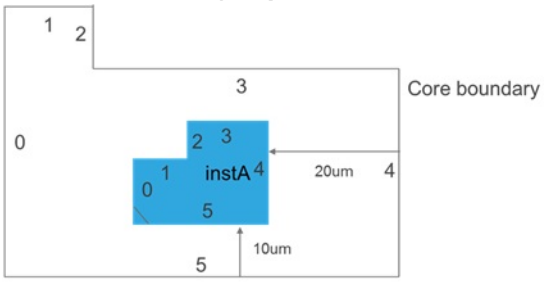
```
create_relative_floorplan -place instA -ref_type object -ref refA \
        -horizontal_edge_separate {1 -20 1} -vertical_edge_separate {2 10 0} -bbox both
```



Placing an Instance According to the Core Boundary

- The following command places an object with respect to the core boundary:

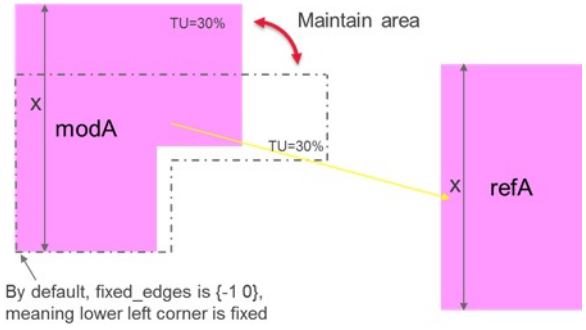
```
create_relative_floorplan -place instA -ref_type core_boundary \
       -horizontal_edge_separate {5 10 5} -vertical_edge_separate {4 -20 4}
```



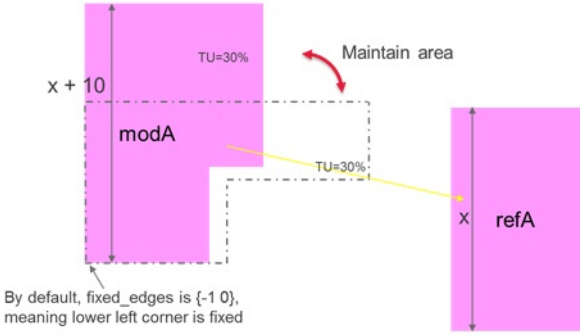
Reshaping an Instance in a Reference Dimension while Maintaining its Current Area

The following commands reshape an object based on a specified height while maintaining its current area:

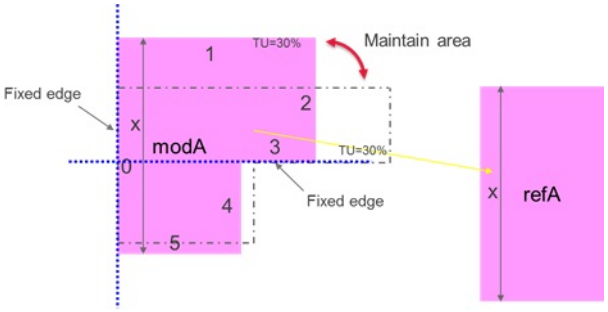
- `create_relative_floorplan -reshape modA -ref refA -dimension height`



- `create_relative_floorplan -reshape modA -ref refA -dimension height -offset 10`

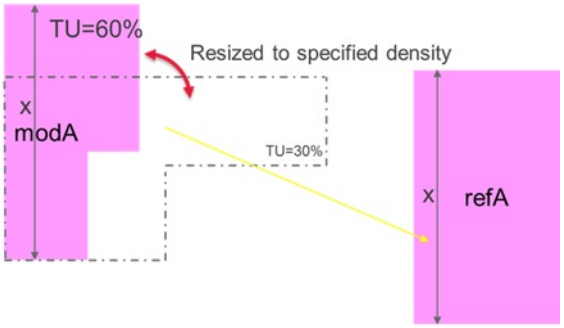


- `create_relative_floorplan -reshape modA -ref refA -fixed_edges {0 3} -dimension height`



Reshaping an Instance while Maintaining a Target Utilization Value

- The following command reshapes an object based on a specified height maintaining a target utilization value:
  `create_relative_floorplan -reshape modA -ref refA  -dimension height -density 0.6`



Placing multiple floorplan objects

- The following command places obj1, obj2, and obj3:

```
create_relative_floorplan -place {obj1 obj2 obj3} -ref {ref1 ref2 ref3} -ref_type {reftype1 reftype2 reftype3} -
horizontal_edge_separate {-1 0 -1,-1 0 -1,-1 0 -1} -vertical_edge_separate {0 -100 2,0 -100 2,0 -100 2} -bbox {both target target} -
orient {MX90 R90 R90}
```

Related Information

- delete_relative_floorplan
- Floorplanning the Design chapter in the *Innovus Stylus Common UI User Guide*

create_resize_blockage

```
create_resize_blockage
[-help]
[-name blockage_name]
[-is_resizeable]
-area x1 y1 x2 y2
```

Adds a size blockage object that controls the behavior of the resize line under a covered area. The object area prevents floorplan objects in this area from being resized and maintains the alignment and minimum space between these objects during floorplan resize. By default resize line can not go through size blockage, which means, objects under size blockage stay fixed after floorplan resize.

Adds a size blockage object that controls the behavior of the resize line under a covered area. The object area prevents floorplan objects in this area from being resized and maintains the alignment and minimum space between these objects during floorplan resize. Size blockage is special kind of blockage that is only honored by the resize_floorplan command. By default resize line can not go through size blockage, which means, objects under size blockage stay fixed after floorplan resize. You can use this command before using the `resize_floorplan` command.

Parameters

`-area x1 y1 x2 y2`	Specifies the lower left x coordinate, lower left y coordinate, upper right x coordinate, and upper right y coordinate respectively of the size blockage area. *Data_type*: rect, required
`-help`	Outputs a brief description that includes type and default information for each `create_resize_blockage` parameter. For a detailed description of the command and all of its parameters, use the man command: `man create_resize_blockage`
`-is_resizeable`	Specifies that the size blockage can be resized, however alignment and the minimum space between the objects in the blockage area will be maintained during floorplan resize. When this options is specified, the resize line can go through size blockage. But it can not turn direction inside blockage. The benefit of using this option is if objects (for example, macros) under size blockage are aligned before calling `resize_floorplan`, they will stay aligned after the command. *Data_type*: bool, optional
`-name blockage_name`	Specifies the name of the size blockage. *Data_type*: string, optional

Example

The following command creates a size blockage.

```
create_resize_blockage -name sizeBlkg1 -area 400 400 500 500
```

Related Information

- `resize_floorplan`
- `resize_floorplan` Category Attributes
- `set_resize_line`
- Floorplan Menu chapter in the *Innovus Stylus Common UI Menu Reference*
- Floorplanning the Design chapter in the *Innovus Stylus Common UI User Guide*
  - Resizing the Floorplan

create_route_blockage

```
create_route_blockage
[-help]
[-fills]
[-inst name]
{-layers layer_name | {layer_name_list ...}  | -all {cut route drc_region trim_metal}}
{-rects {{x1 y1 x2 y2} ...} | -polygon {{x1 y1} {x2 y2} ...} | -cover }
[-mask integer]
[-no_wrong_way]
[-name blk | -use_prefix ]
[-except_pg_nets | -pg_nets ]
[-except_pg_nets | -pg_nets | -partial integer]
[-spacing float | -design_rule_width float]
```

Creates a routing blockage object that can be moved even outside the core area. The object area prevents routing of specified metal layers, signal routes, and hierarchical instances in this area. Use this command during partition floorplanning.

Parameters

`-all {cut route drc_region trim_metal}`	
	Provides a quick way to specify layer types if all layers of the type are to be blocked. You can specify multiple layer types with this param • `cut`: Specify to create blockages on all cut layers. • `route`: Specify to create blockages on all routing layers. • `drc_region`: Specify to create routing blockages on drc region layers. • `trim_metal`: Specify to create routing blockages on all trim metal layers. *Data_type*: (enum)+, optional
`-cover`	Specifies that a routing blockage of the same size as the specified block instance (`-inst name`) will be created on top of the instance. Th *Data_type*: bool, optional
`-design_rule_width float`	
	Specifies that the blockage has a width of *effective_width* for the purposes of spacing calculations. If you specify `-design_rule_width`, **Type:** The *effective_width* is a real number greater than 0, in microns. *Data_type*: float, optional
`-except_pg_nets`	Specifies that the routing blockage is to be applied on a signal net routing and not on power or ground net routing. Use this option to blo allow power connections to go through the blockage. **Note:** Blocking the signal net routing helps in avoiding crosstalk or coupling caused by signal routes. *Data_type*: bool, optional
`-fills`	Specifies that the routing blockage is to be applied on metal fills. *Data_type*: bool, optional
`-help`	Prints a brief description that includes the type and default information for each `create_route_blockage` parameter. For a detailed descri `man create_route_blockage`
`-inst name`	Specifies the name of the instance on which the routing blockage is to be applied. **Note:** When the `-cover` parameter is specified, the `-inst` parameter must also be specified. However, when the `-inst` parameter is spec *Data_type*: inst, optional
`-layers layer_name | {layer_name_list ...}`	

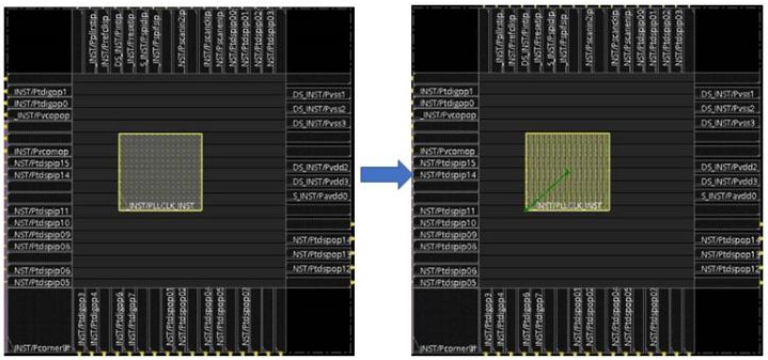
	Specifies the layer or list of layers on which the routing blockage is to be applied. The allowed layers are routing layer, cut layer, drc regi **Note:** The values specified with this parameter will override any default blockage layers specified with the `set_route_blockage_default` *Default*: If you do not specify this parameter, the Innovus software will check if any default blockage layers have been specified with the `set_route_blockage_default_layer` command will be used. If no, the default is metal3 if the number of layers is greater than 3, or metal2 **Note**: You must specify the *layer_name* as a number or as a layer name defined in LEF. For example, *layer_name* can be specified as 2 *Data_type*: (layer)+, optional
`-mask integer`	Blocks routing layers with specified masks. Layers with other masks can still be routed through the blockage. **Note:** • The default value of the `-mask` parameter is 0. This implies that all the specified layers will be blocked. • If the `-mask` parameter is specified with an integer value greater than 0, then both layers with specified mask and uncolored layers are Example. If `-mask 1` is specified, it blocks mask 1 and uncolored routes. • If more than one layers are associated with a mask, the mask is applied to all of them. • If the `-mask` parameter is specified, the tool checks if the mask is suitable for all of the specified layers. If not, it displays an ERROR m • The NanoRoute router and the `check_drc` command honor the `-mask` parameter. *Default*: 0 *Data_type*: int, optional
`-name blk`	Specifies the layer name of the route blockage. *Data_type*: string, optional
`-no_wrong_way`	When this parameter is specified, wrong way routing is avoided in the region. Both router and verify_drc will report DRC if any jogs are p *Data_type*: bool, optional
`-partial integer`	Specifies the density for partial routing blockages. It causes the global router to only use up to the specified percent of the routing resour routes on that layer in the area. For example, `-partial 75` means only 75% of routing resource in a gcell is available for global routing, and this will affect the final dema This can be useful if the router consistently creates too much congestion in a specific area. If you add a partial routing blockage in that a overlapping partial routing blockages, the lowest density percentage of the blockages will be used in the overlapping region. For examp routing blockage with 50 percent density, the density of the overlapping region will be 50 percent. **Note:** You can attach a partial routing blockage to a block instance with `-inst` if the congestion is related to the boundary and pins of a s **Note:** The `-design_rule_width`, `-except_pg_nets`, `-pg_nets`, and `-spacing` parameters are not allowed for partial routing blockages. *Minimum*: 0 *Maximum*: 100 *Data_type*: int, optional
`-pg_nets`	Specifies that the routing blockage is to be applied only on power or ground net special routes and not on signal nets. This option affects when they are used for PG nets. It does not affect signal routing of PG nets done by `route_early_global`. NanoRoute, which is used to c connections, is also not affected by this option. Use this option during power planning to prevent power routes from getting too close to b channels. **Note:** This option cannot be used with `-except_pg_nets`. *Data_type*: bool, optional
`-polygon {{x1 y1} {x2 y2} ...}`	
	Specifies the polygon vertices of the blockage, in microns. **Note:** Although `create_route_blockage` can be be used to create triangular blockages, metal fill does not currently support triangular rou *Data_type*: (point)+, optional
`-rects {{x1 y1 x2 y2} ...}`	

	Specifies the coordinates of a list of blockage areas.
	Data_type: (rect)+, optional
	For example, the following command creates a routing blockage object on metal layers 1, 4, and 5:
	`create_route_blockage -rects {0 0 100 100} -layer {metal1 metal4 metal5}`
	Note: You can also add variables to specify the blockage area. For example,
	`set box "0 0 100 100"`
	`create_route_blockage -rects $box`

`-spacing` *float*	Specifies the minimum spacing allowed between the blockage and any other routing shape. If you specify `-spacing`, you cannot specify T**ype:** The float is a real number greater than or equal to 0, in microns. *Data_type*: float, optional
`-use_prefix`	Specifies that the routing blockages should be named with a prefix. *Data_type*: bool, optional

Example

- The following command creates a routing blockage object to block metal3 routing in this area:
  `create_route_blockage -rects {2353.8000 7123.000 2653.8000 8298.2000} -layers metal3`
- The following command creates a routing blockage object on metal layers 1, 4, and 5:
  `create_route_blockage -rects {0 0 100 100} -layers {metal1 metal4 metal5}`
- The following command creates a routing blockage object on metal layers 1, 4 and blocks the signal net routing:
  `create_route_blockage -rects {10 18 1782 2300} -layers {metal1 metal4} -except_pg_nets`
- The following command creates a routing blockage object having polygon vertices (10, 10), (10, 15), (15, 15), (15, 20), (20, 20), (20, 10) on the metal layer 1:
  `create_route_blockage -polygon {10 10 10 15 15 15 15 20 20 20 20 10} -layers metal1`
- The following command creates a routing blockage object on top of the the specified block instance that is of the same size the instance.
  `create_route_blockage -inst DTMF_INST/PLLCLK_INST -cover`



- The following command creates a routing blockage object all the layers:
  `create_route_blockage -name inist1_fill_blk -fills -rects [get_db [get_db insts inst_U1] .place_halo_bbox] -layers *`
- The following command creates a routing blockage on metal fills only:
  `create_route_blockage -rects {500 400 600 500} -layers metal1 -fills`
- The following command creates a routing blockage on power or ground net special routes only and not on signal nets.
  `create_route_blockage -rects {0 0 100 100} -layers metal1 -pg_nets`

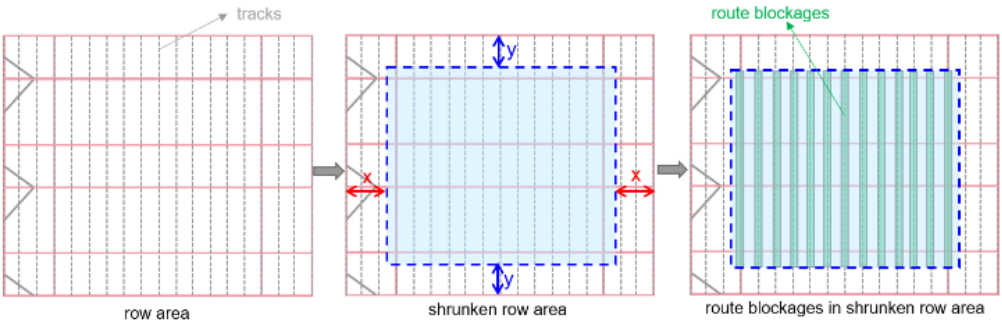
Related Information

- `delete_route_blockages`
- `set_route_blockage_default_layer`
- Floorplanning the Design chapter in the *Innovus Stylus Common UI User Guide*

create_route_blockage_along_track

```
create_route_blockage_along_track
[-help]
-align_track string
[-consider_cells_as_macros string]
[-consider_macros_as_cells string]
[-edge_shrink {x y}]
-layer string
[-no_cut_row]
```

Creates routing blockages that are aligned to specified tracks which are in the shrunken row area. Use this command during partition floorplanning.

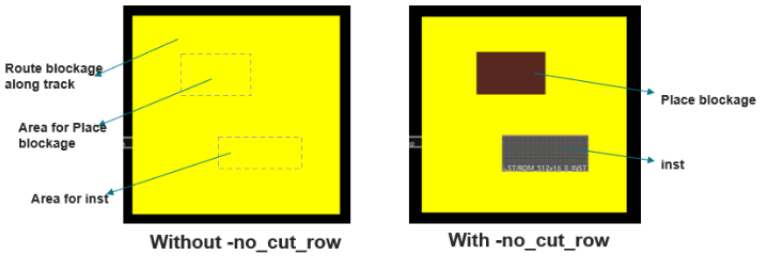


Note:

- Routing blockages are also created with align to the track which is near to the shrunken row area with less than half of layer width.
- Routing blockages honor rectilinear row area.

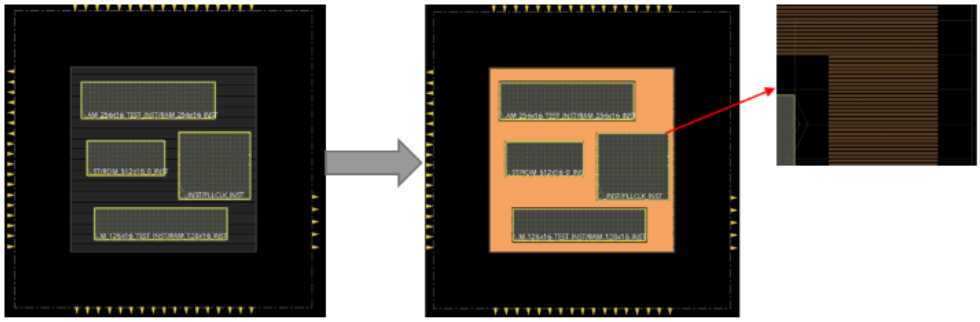
Parameters

`-align_track string`	Specifies the track layer to which the route blockages align to. *Data_type*: string, required
`-help`	Prints a brief description that includes the type and default information for each `create_route_blockage_along_track` parameter. For a detailed description of the command and all of its parameters, use the man command: `man create_route_blockage_along_track`
`-consider_cells_as_macros string`	
	Specifies the names of the cells that are considered as macros. Blockages are not created over them. *Data_type*: string, optional
`-consider_macros_as_cells string`	
	Specifies the names of the cells that are considered as std cells. Blockages are created over them. *Data_type*: string, optional
`-edge_shrink {x y}`	Shrinks row area from the edge in X and Y direction. *Data_type*: point, optional
`-layer string`	Specifies the name of the layer on which the tool creates routing blockages for steam out. *Data_type*: string, required

`-no_cut_row`	Specifies whether the blockages to be added should be based on the no-cut row site area even if the row site area is not cut in the design. • Without the `-no_cut_row` parameter, the tool will create route blockages based on the current row area. So the rows for objects like insts, place blockages are expected to be cut out before running the `create_route_blockage_along_track` command to get expected results. • With the `-no_cut_row` parameter, the tool will identify the rows have not been cut before and will call the `split_row` command to get a cut row shape during the action. The route blockage is created based on the cut row shape. The actual row shape will not be changed. • The f`loorplan_no_cut_row true` attribute has the same use with this option. Example: In the following illustration, the rows are not cut before. Now, without the `-no_cut_row` parameter, the tool will create a route blockage over the place blockage and the inst. With the `-no_cut_row` parameter, the tool will not create route blockage over them.  *Default:* false *Data_type*: bool, optional

Example

- The following command creates a routing blockage on layer M2. The route blockages align to track layer M2:
  `create_route_blockage_along_track -layer M2 -align_track M3 -edge_shrink {5 5}`



Related Information

- `delete_route_blockages`
- `set_route_blockage_default_layer`
- Floorplanning the Design chapter in the *Innovus Stylus Common UI User Guide*

create_route_halo

```
create_route_halo
[-help]
{ {-bottom_layer bottom_layer -space halo_value -top_layer top_layer} | -litho_halo }
{-all_blocks | -block block_name_list | -cells cell_name_list | -insts instance_name | -design_halo }
```

Adds a routing halo for blackboxes, hard macros, or block-level designs. A routing halo significantly reduces the possibility of long wire routing within the specified area, thus helping reduce signal integrity violations at the top- and/or block-level designs. A routing halo is honored only by the signal router. The signal router treats routing on the specific routing layers in the routing halo area as very high cost routing. However, perpendicular routing or straight connections to pins are acceptable. The special router does not honor routing halos. To block special routing, use routing blockages instead. You can specify a routing halo for:

- hard macros or blackboxes
- block-level designs
- partitions

For the bottom-up hierarchical flow, use this command to specify routing halo for a block at the top-level design or at the block-level design. The routing halo can be stored as an instance and/or design property and can be viewed through the attribute editor for the instance block. The routing halo can be saved to or restored from a floorplan file. Use this command after importing a design.

Notes:

- For top-down hierarchical flow, specify routing halo for a partition through the `create_partition` command or through the *Specify Partition* form.
- A halo is not allowed on std-cells (cells with CLASS CORE SITE) and is written out as ROUTEHALO in DEF. A design halo is written out as a DEF design FE_ROUTING_HALO property and a litho halo is written out as a DEF component LITHOHALO property.

Parameters

`-all_blocks`	Specifies that the routing halo should be added to all blocks in the design. *Data_type*: bool, required
`-block` *block_name_list*	Specifies that the routing halo should be added to the specified blocks. A block name must be an instance name and not a cell master name. The specified blocks should be separated by a space. *Data_type*: list, required
`-bottom_layer` *bottom_layer*	Specifies the bottom layer for the routing halo. **Note**: Even though the `-bottom_layer`, `-space`, and `-top_layer` parameters are optional, it's important to specify all of them together. *Data_type*: string, required
`-cells` *cell_name_list*	Specifies that the routing halo should be added to all insts of the specified cell names. *Data_type*: list, required
`-design_halo`	Specifies that the routing halo should be added at the block-level design. *Data_type*: bool, required
`-help`	Outputs a brief description that includes the type and default information for each `create_route_halo` parameter. For a detailed description of the command and all of its parameters, use the man command: `man create_route_halo`
`-insts` *instance_name*	Specifies that the routing halo should be added to the specified instance. This parameter can be useful for adding halo for I/O cells. *Data_type*: list, required
`-litho_halo`	Adds a litho halo to the instances on each routing layer with `LEF LITHOMACROHALO` defined. **Note**: The `-litho_halo` parameter cannot be specified with the `-design_halo` parameter. These parameters are mutually exclusive. **Note**: The `-litho_halo` parameter requires some `LEF LITHOMACROHALO` values to be set. **Note**: A halo is not allowed on std-cells (cells with CLASS CORE SITE) and is written out as ROUTEHALO in DEF. A design halo is written out as a DEF design FE_ROUTING_HALO property and a litho halo is written out as a DEF component LITHOHALO property. *Data_type*: bool, required
`-space` *halo_value*	Specifies the routing halo value in microns. *Minimum*: 0 *Maximum*: 1.79769313486232e+308 *Data_type*: float, required
`-top_layer` *top_layer*	Specifies the top layer for the routing halo. *Data_type*: string, required

Examples

- The following command adds a routing halo of 10 microns to all blocks and specifies that the top and the bottom layers for the routing halo are Metal7 and Metal1, respectively.
  ```
  create_route_halo -space 10 -top_layer Metal7 -bottom_layer Metal1 -all_blocks
  ```
- The following command adds a routing halo of 5 microns to the specified blocks and specifies that the top and the bottom layers for the routing halo are Metal2 and Metal1, respectively.
  ```
  create_route_halo -block DTMF_INST/RAM_256x16_TEST_INST/RAM_256x16_INST DTMF_INST/RAM_128x16_TEST_INST/RAM_128x16_INST -space 5 -top_layer Metal2 -bottom_layer Metal1
  ```

Related Information

- `delete_route_halos`
- `create_partition`
- Floorplanning the Design chapter in the *Innovus Stylus Common UI User Guide*
- Floorplan Menu chapter in the *Innovus Stylus Common UI Menu Reference*
  - Edit Halo

create_route_rule

```
create_route_rule
[-help]
[-add_via {via_name1 via_name2 ...}]
[-exclude_backside_via]
[-generate_via]
[-hard_spacing]
[-init string]
[-min_cut {layer1[:layer2] min_cut ...}]
[-use_via_cut_class string]
[-spacing {layer1[:layer2] spacing ...} |
 -spacing_multiplier {layer1[:layer2] spacing_multiplier}]
[-via {via_name1 via_name2 ...}]
[-width {layer1[:layer2] width ...} |
 -width_multiplier {layer1[:layer2] width_multiplier ...}]
{-name rule_name}
```

Adds a non-default rule for objects. You can also remove vias from the NDR. The command checks the `add_route_via_defs` mode and if the `add_route_vias_auto` attribute is set to `true`, the generated vias is used irrespective of the `-generate_via` option. You can use the command at any point in the design flow.

Parameters

`-help`	Prints a brief description that includes type and default information for each `create_route_rule` parameter. For a detailed description of the command and all of its parameters, use the man command: `man create_route_rule`
`-add_via {`*`via_name1 via_name2 ...`*`}`	
	Adds vias to the existing via list. You can use this parameter to add DFM vias or other custom vias. *Data_type*: string, optional
`-exclude_backside_via`	Prevents inclusion of non-route or backside vias in a newly defined non-default rule. *Data_type*: bool, optional
`-generate_via`	
	Auto-generates vias with the `add_route_via_defs` command. **Note:** If the NDR specified by you is with metal wider than the default rule, it is recommended to use this option for getting better routing results. *Data_type*: bool, optional
`-hard_spacing`	Sets the non-default rule as HARDSPACING. *Data_type*: bool, optional
`-init` *`string`*	Specifies the name of an existing non-default rule from which all values are initialized. **Note:** Other options will overwrite these values. *Data_type*: string, optional
`-min_cut {`*`layer1`*`[:`*`layer2`*`] `*`min_cut ...`*`}`	
	Specifies the minimum cut limit for via layers. *Data_type*: string, optional
`-name` *`rule_name`*	Specifies the name of the non-default rule that is to be added. *Data_type*: string, optional
`-spacing {`*`layer1`*`[:`*`layer2`*`] `*`spacing ...`*`}`	
	Specifies the spacing for routing layers. *Data_type*: string, optional
`-spacing_multiplier {layer1[:layer2]` *`spacing_multiplier`*`}`	
	Spacing multiplier for routing layers. *Data_type*: string, optional
`-use_via_cut_class` *`string`*	Specifies the layer and `cut_class` list to support the LEF/DEF property. *Data_type*: string, optional
`-via {`*`via_name1 via_name2 ...`*`}`	
	Resets and adds the list of via or via rule generate name. *Data_type*: string, optional
`-width {`*`layer1`*`[:`*`layer2`*`] `*`width ...`*`}`	
	Specifies the width for routing layers. *Data_type*: string, optional
`-width_multiplier {`*`layer1`*`[:`*`layer2`*`] `*`width_multiplier ...`*`}`	
	Width multiplier for routing layers. *Data_type*: string, optional

Example

- The following command creates a rule that requires extra spacing. All other values are the same as the default rule.
  ```
  create_route_rule -name default_2x_space -spacing {metal1:metal4 0.2 metal5:metal6 0.4 metal7 0.6}
  ```
- The following command creates a new wide-wire routing rule (double wide on the Mx routing layers, but not the wide layers). Auto-generate vias because the width changed, so the default vias are not correct.
  ```
  create_route_rule -name Mx_2x_wide -width {metal1:metal4 0.20} -generate_via
  ```
- The following command uses existing wide-wire rule, but require 2 cut vias for the Vx layers (via1 to via3)
  ```
  create_route_rule -name Mx_2x_wide_Vx_2cut -init Mx_2x_wide -min_cut {via1:via3 2}
  ```

Related Information

- `add_route_via_defs`
- Using the NanoRoute Router chapter in the *Innovus Stylus Common UI User Guide*
  - Using Non-Default Rules

create_row

```
 create_row
[-help]
-site siteName
[-flip_first_row]
[-limit_rows_in_core]
[-no_check]
[-no_flip_rows]
[-spacing distance]
[-no_abut | -no_abut_first ]
[-area {x1 y1 x2 y2} | -rects {{x1 y1} {x2 y2}...} | -polygon {{x1 y1} {x2 y2}...}]
```

Creates rows for the specified site. The row boundary can be defined by core area or the area that you specify. This command supports the creation of overlapping rows. This command can create only horizontal rows. By default, alternate rows are flipped and all rows are abutted (no spaces between rows).

The following points apply to the usage of this command:

- Rows can be created both inside and outside the core area, but within the die.
- All new rows created in an area must be an integer multiple of any existing rows in the same area.
- Non-integer multiple height rows cannot be created outside power domains.

Use this command after importing a design.

Parameters

`-area {x1 y1 x2 y2}`	
	Create rows in the specified rectangle. (Default is within the core box.) The values are floats, units are microns. Rows are truncated at the die box boundary. Either a tcl list or 4 separate parameters works. • *x1* : Lower left x coordinate. • *y1* : Lower left y coordinate. • *x2* : Upper right x coordinate. • *y2* : Upper right y coordinate. *Data_type*: rect, optional
`-flip_first_row`	Controls the orientation of the first (bottom) row, which is R0 by default. When `-flip_first_row`, the first row's orientation is MX. See `-no_flip_rows`. *Data_type*: bool, optional
`-help`	Outputs a brief description that includes the type and default information for each `create_row` parameter. For a detailed description of the command and all of its parameters, use the man command: `man create_row`
`-limit_rows_in_core`	By default (none of `-area, -polygon, -rects` specified) rows are created in the core box. When one of `-area, -polygon, -rects` is specified, by default rows are created within the intersection of the die box and the specified rectilinear shape. When one of `-area, -polygon, -rects` and `-limit_rows_in_core` is specified, rows are created within the intersection of the core box and the specified rectilinear shape. *Data_type*: bool, optional
`-no_abut`	Controls where spaces are inserted between rows. By default, spaces are inserted between pairs of rows. When `-no_abut` is enabled spaces are inserted between every row. See `-spacing` and `-no_abut_first`. *Data_type*: bool, optional
`-no_abut_first`	Controls where the first space between rows is inserted. Default is that the first space is inserted after the first (lowest) *pair* of rows, and subsequently after every pair of rows (space inserted between 2nd and 3rd rows, between 4th and 5th rows, etc). When `-no_abut_first`, the first space is inserted after the first (lowest) row, and subsequently after every pair of rows (space inserted between 1st and 2nd rows, then 3rd and 4th rows, etc). See `-spacing` and `-no_abut`. *Data_type*: bool, optional

`-no_check`	Specifies that the row should not be aligned to the existing row and should support redo undo. *Data_type*: bool, optional
`-no_flip_rows`	By default rows have alternating orientation `(...MX R0 MX R0...)`, or when `-no_flip_rows` is specified rows have identical orientation (one of: `...R0 R0 R0...`; or `...MX MX MX...`). See `-flip_first_row`. *Data_type*: bool, optional
`-polygon {{x1 y1} {x2 y2}...}`	
	The X and Y values are floats, units are microns. There must be an even number of points specified. The points specify a polygon in which rows are created. Each pair of points specifies one edge of the polygon, so { X1 Y1 } and { X2 Y2 } specify one edge, { X2 Y2 } { X3 Y3 } specify the next edge, etc. The last edge is specified by { Xlast Ylast } { X1 Y1 }. In the degenerate case (multiple rectilinear areas touching at corners) rows will be created continuously starting at the lowest point. *Data_type*: (point)+, optional
`-rects {{x1 y1} {x2 y2} ...}`	
	The X and Y values are floats, units are microns. There must be an even number of points specified. Pairs of points are used to determine a set of rectangles, where the first point of the pair is LLX, LLY, the second point is URX, URY. Rectangles need not be overlapping. Rows are created in the union of the rectangles defined. Note that for overlapping rectangles, rows are created continuously starting at the first (bottom) row (of the lowest rectangle). For non-overlapping rectangles, rows are created independently starting at the bottom of each rectangle. *Data_type*: (point)+, optional
`-site` *siteName*	Specifies the name of the site to be used for creating the rows. (Objects not allowed.) *Data_type*: string, optional
`-spacing` *distance*	Specifies the width of the space inserted between certain rows. The first (lowest) row is always inserted along the bottom edge of the area having rows added. If `-no_abut` is specified, then space is inserted between every subsequent row. The default is space is inserted between every other row. If `-no_abut_first` is specified, then space is inserted after the first row (and then the third etc). The default is that space is inserted after the second row (and then the 4th etc). Note that `-no_abut` and `-no_abut_first` are mutually exclusive. *Default*: 0.0 um *Data_type*: float, optional

Example

The following command creates rows for the site lt3site within the area defined by the coordinates 376.7, 577.57, 688.94, 725.68. A 1 micron space is inserted between the 1st (lowest) and 2nd rows, the 3rd and 4th rows, etc:

```
create_row -site lt3site -area 376.7 577.57 688.94 725.68 -spacing 1.0 -no_abut_first
```

Related Information

- `delete_row`
- `init_core_rows`
- `split_row`
- `stretch_rows`
- Floorplanning the Design chapter in the *Innovus Stylus Common UI User Guide*
  - Using Vertical Rows

create_shape

```
create_shape
[-help]
-layer {layer_name_or_pointer}
[-net {net_name}]
[-shape {none ring stripe followpin iowire corewire blockwire fillwire blockagewire padring blockring drcfill fillwireopc}]
[-shield_net {net_name_or_pointer}]
[-status {routed fixed cover shield noshield}]
[-user_class className]
{-rect {x1 y1 x2 y2} | -patch {x1 y1 x2 y2} | -polygon {x1 y1 x2 y2 ...xn yn} |
[-path_segment {x1 y1 x2 y2} -width value
 [-begin_extension value -end_extension value]]}
```

The command `create_shape` adds `DEF SPECIALNETS` wiring shapes to the database. You can use the command at any point in the design flow.

Parameters

`-help`	Prints a brief description that includes type and default information for each `create_shape` parameter. For a detailed description of the command and all of its parameters, use the man command: `man create_shape`
`-begin_extension value`	Specify the extension beginning value in microns. *Default*: 0 *Data_type*: float, optional
`-end_extension value`	Specify the extension end value in microns. *Default*: 0 *Data_type*: float, optional
`-layer {layer_name_or_pointer}`	
	Name of the layer or layer pointer. *Data_type*: string, required
`-net net_name`	Name of the net. *Data_type*: string, optional
`-patch {x1 y1 x2 y2}`	Specifies the coordinates of the lower-left and upper-right points of the rectangle for creating the patch wires. *Data_type*: rect, optional
`-path_segment {x1 y1 x2 y2}`	
	The beginning and ending coordinates for the path segment. **Note:** For vertical stripes, x1 and x2 are the same (and are the coordinates of the center of the stripe). Similarly for horizontal stripes. *Data_type*: rect, optional
`-polygon {x1 y1 x2 y2 ...xn yn}`	
	The point list of the polygon, in microns. Note: add shapes to signal nets with `-path_segment` *Data_type*: (point)+, optional
`-rect {x1 y1 x2 y2}`	
	Specifies the coordinates of the lower-left and upper-right points of the rectangle for creating horizontal and vertical wires. *Data_type*: rect, optional
`-shape {none ring stripe followpin iowire corewire blockwire fillwire blockagewire padring blockring drcfill fillwireopc}`	

	Defines the `specialnets` wiring shape. *Default*: `None` *Data_type*: enum, optional
`-shield_net {`*net_name_or_pointer* `}`	
	The name or pointer of the net to be shielded by the via instance. The `-shield_net` option is only allowed when `-status shield` is used. *Data_type*: string, optional
`-status {routed fixed cover shield noshield}`	
	Defines the `specialnets` wiring status. *Default*: `routed` *Data_type*: enum, optional
`-width` *value*	The width in microns. *Data_type*: float, optional
`-user_class`	Defines the attribute class by user. *Data_type*: string, optional

Example

- `-path_segment`: The following command adds a horizontal `pathSeg` to the net with the name `VSS` on layer `METAL1` at the co-ordinates (100, 250, 200 ,250) with a width of 2.5 microns and the begin and end extensions of 1.25 microns.
  Note: Typically, the begin and end extension values are zero for most `DEF SPECIALNETS` wiring, but can be any value that would result in the end of the wire falling on the manufacturing grid.
  ```
  create_shape -net VSS -layer METAL1 -path_segment {100 250 200 250} -width 2.5 -begin_extension 1.25 -end_extension 1.25
  ```
- `-polygon`: This command adds a polygon shaped wire to the net named `VSS` on layer `METAL1` with coordinates specified.
  ```
  create_shape -net VSS -layer METAL1 -polygon {{25.05 25.115} {25.115 25.05} {25.115 24.95} {5.05 4.885} {4.95 4.885} {4.885 4.95} {4.885 5.05} {24.95 25.115}}
  ```
- This command would add a polygon shape swire to snet `VSS` on layer `METAL3` having coordinates similar to the swire belonging to the net `VSS` on layer `METAL1`.
  ```
  create_shape -net VSS -layer METAL3 -polygon [ get_db [ get_db [ get_db nets VSS ] .special_wires -if { .layer.name == METAL1 } ] .polygon ]
  ```

See also:

- `select_routes`
- `delete_obj`
- `add_stripes`
- DEF Syntax chapter in the *LEF/DEF Language Reference*

create_special_drc_region

```
  create_special_drc_region
[-help]
[-extend_outer_x float]
[-extend_outer_y float]
[-extend_x float]
[-extend_y float]
[-insts string]
-layer string
```

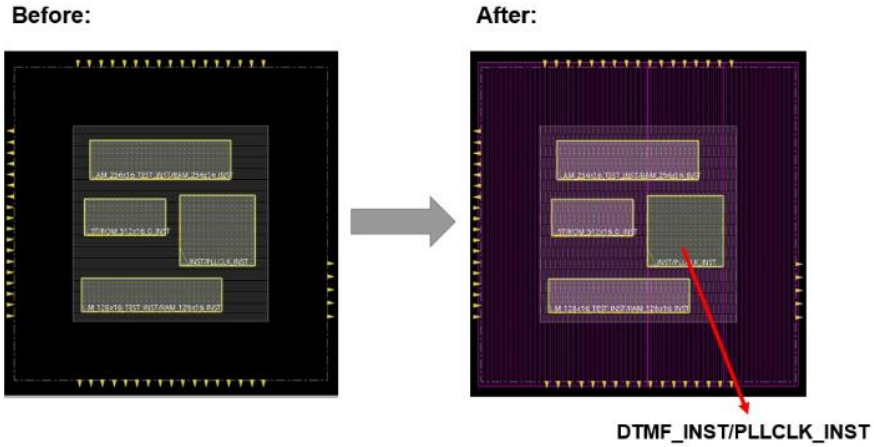
DRC region layers define a special masterslice layer that is used to define the areas of a region on which a set of rules defined in the metal, cut, and/or trim metal layers with the REGION property would be applied. The `create_special_drc_region` command can be used to add drc region blockages in or around the die area except specified macros.

Parameters

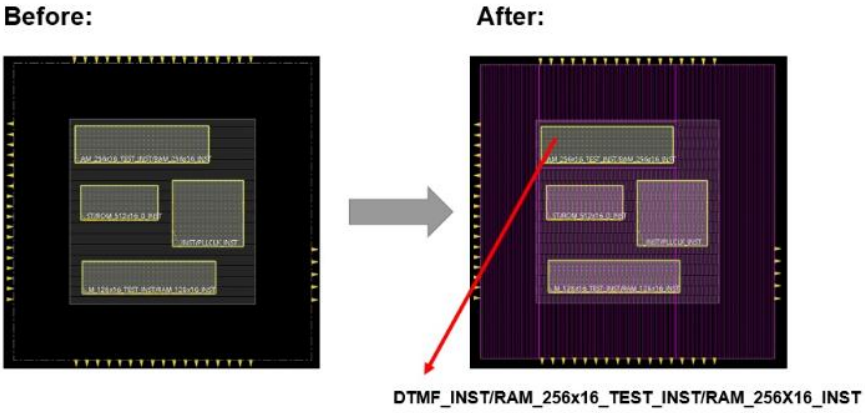
`-extend_outer_x float`	Expands the routing blockage in X direction. *Data_type*: float, optional
`-extend_outer_y float`	Expands the routing blockage in Y direction. *Data_type*: float, optional
`-extend_x float`	Expands the non-routing blockage in X direction. *Data_type*: float, optional
`-extend_y float`	Expands the non-routing blockage in X direction. *Data_type*: float, optional
`-help`	Outputs a brief description that includes the type and default information for each `create_special_drc_region` parameter. For a detailed description of the command and all of its parameters, use the man command: `man create_special_drc_region`
`-insts string`	Specifies the name of the hierarchical instance that are to be cut out. **Note:** Wild card characters are not supported while specifying hierarchical instance names. **Note:** All blocks are included if `-insts` is not specified or when `-insts{}` is specified. *Data_type*: string, optional
`-layer string`	Specifies the names of the layers on which the routing blockage is created. *Data_type*: string, optional

Example

- The following command creates a routing blockage on R1 that blocks out `DTMF_INST/PLLCLK_INST`:
  `create_special_drc_region -layer R1 -extend_outer_x 10 -extend_outer_y 20 -insts DTMF_INST/PLLCLK_INST`



- The following command creates a routing blockage on R1 that blocks out `DTMF_INST/RAM_256x16_TEST_INST/RAM_256x16_INST`:
  `create_special_drc_region -layer R1 -extend_x 10 -extend_y 20 -insts DTMF_INST/RAM_256x16_TEST_INST/RAM_256x16_INST`



Related Information

- Floorplanning the Design chapter in the *Innovus Stylus Common UI User Guide*

create_special_mark_layer_shape

```
create_special_mark_layer_shape
[-help]
[-cut_out_block_insts string]
[-extend_outer_x float]
[-extend_outer_y float]
[-extend_x float]
[-extend_y float]
-layer string
```

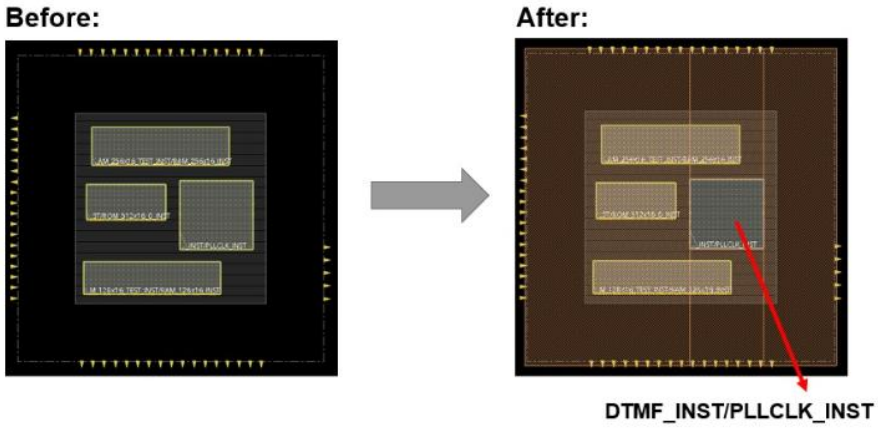
Creates a mark layer shape, using special route. This command enables you to apply one set of rules for the entire design in Innovus by connecting the macro pins around halo of IP blocks. The IPs are selected based on a list which provides the instance name.

Parameters

`-cut_out_block_insts string`	Specifies the name of the instances that are to be cut out. **Note:** All blocks are cut out if `-cut_out_block_insts` is not specified however, if `-cut_out_block_insts {}` is specified then all blocks are NOT cut as this pattern cannot match anything. *Data_type*: string, optional
`-extend_outer_x float`	Expands the mark layer shape in X direction. *Data_type*: float, optional
`-extend_outer_y float`	Expands the mark layer shape in Y direction. *Data_type*: float, optional
`-extend_x float`	Expands the no-mark layer shape in X direction. *Data_type*: float, optional
`-extend_y float`	Expands the no-mark layer shape in Y direction. *Data_type*: float, optional
`-help`	Outputs a brief description that includes the type and default information for each `create_special_mark_layer_shape` parameter. For a detailed description of the command and all of its parameters, use the man command: `man create_special_mark_layer_shape`
`-layer string`	Specifies the names of the layers on which the mark layer shape is to be applied. *Data_type*: string, required

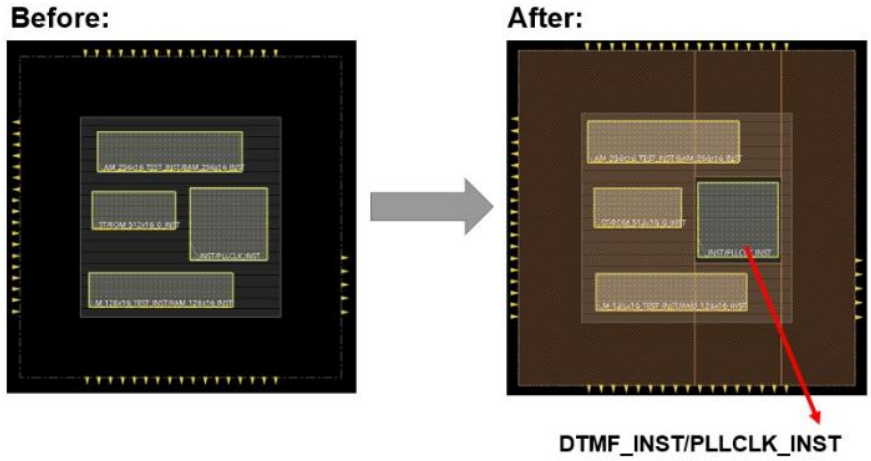
- The following command creates a mark layer shape on M2_P48 that blocks out `DTMF_INST/PLLCLK_INST` and expands the marker layer shape by 10 in X direction and by 20 in Y direction:

  ```
  create_special_mark_layer_shape -cut_out_block_insts DTMF_INST/PLLCLK_INST -extend_outer_x 10 -extend_outer_y 20 -layer M2_P48
  ```



- The following command creates a mark layer shape on M2_P48 that blocks out `DTMF_INST/PLLCLK_INST` and expands the no-mark layer shape by 10 in X direction and by 20 in Y direction:

```
create_special_mark_layer_shape -cut_out_block_insts DTMF_INST/PLLCLK_INST -extend_x 10 -extend_y 20 -layer M2_P48
```



Related Information

- Floorplanning the Design chapter in the *Innovus Stylus Common UI User Guide*

create_text

```
create_text
[-help]
[-alignment {center_center center_left center_right lower_center lower_left
             lower_right upper_center upper_left upper_right}]
[-always_display {true | false}]
[-font {euroStyle gothic math roman script stick fixed swedish milSpec}]
[-height value]
-label {string}
[-layer {layer_name_or_pointer}]
[-oa_purpose {string}]
[-orient {mx mx90 my90 r0 r180 r270 r90}]
-point {x y}
```

Adds text to the specified custom layer. You can use the `create_text` command at any point in the design flow. The text added by the command is to annotate comment text and does not show up in a `DEF` output.

Parameters

`-help`	Prints a brief description that includes type and default information for each `create_text` parameter. For a detailed description of the command and all of its parameters, use the man command: `man create_text`
`-alignment {center_center center_left center_right lower_center lower_left lower_right upper_center upper_left upper_right}`	
	Specifies the alignment of the text. *Data_type*: enum, optional
`-always_display {true | false}`	
	Indicates if the text is always displayed. *Data_type*: enum, optional
`-font {euroStyle gothic math roman script stick fixed swedish milSpec}`	
	Specifies the font style of the text. *Default*: `stick` *Data_type*: enum, optional
`-height` *value*	Specifies the height of the text, in microns. *Data_type*: float, optional
`-label {`*string*`}`	
	Specifies the text string. *Data_type*: string, required
`-layer {`*layer_name_or_pointer*`}`	
	Specifies the layer name or the layer pointer. *Data_type*: string, optional
`-oa_purpose {`*string*`}`	Specifies a user-defined purpose name for OpenAccess text layer purpose pair support. Only values that exist in the library's tech graph are allowed. If the specified name is not a legal purpose name, a warning is displayed and the text is treated as drawing purpose. *Default:* `drawing` *Data_type*: string, optional
`-orient {mx mx90 my90 r0 r180 r270 r90}`	
	Specifies the orientation of the text. *Data_type*: enum, optional
`-point {`*x y*`}`	Specifies the beginning coordinates of the text. It is specified in microns. *Data_type*: point, required

Related Information

- Floorplanning the Design chapter in the *Innovus Stylus Common UI User Guide*

delete_all_density_area

```
delete_all_density_area
[-help]
```

Removes all partial placement blockages from the floorplan. Use this command after importing the design.

Parameters

None

`-help`	Outputs a brief description of the `delete_all_density_area` command. For a detailed description of the command, use the man command: `man delete_all_density_area`

Related Information

- Floorplanning the Design chapter in the *Innovus Stylus Common UI User Guide*

delete_all_floorplan_objs

```
delete_all_floorplan_objs
[-help]
```

Removes all floorplan objects. The density screen, obstruction, power strips, and route guides objects are deleted. The guides and block guides are cleared from the design area. The physical instance groups are also cleared from the core area. Use this command after importing the design.

Parameters

None

`-help`	Outputs a brief description of the `delete_all_floorplan_objs` command. For a detailed description of the command, use the man command: `man delete_all_floorplan_objs`

Related Information

- `delete_selected_from_floorplan`
- Floorplanning the Design chapter in the *Innovus Stylus Common UI User Guide*
- Floorplan Menu chapter in the *Innovus Stylus Common UI Menu Reference*
  - Clear Floorplan

delete_all_power_preroutes

```
delete_all_power_preroutes
[-help]
```

Removes all power preroutes from the floorplan. Use this command after importing the design.

Parameters

None

`-help`	Outputs a brief description of the `delete_all_power_preroutes` command. For a detailed description of the command, use the man command: `man delete_all_power_preroutes`

Related Information

- Floorplanning the Design chapter in the *Innovus Stylus Common UI User Guide*

delete_all_signal_preroutes

```
delete_all_signal_preroutes
[-help]
```

Removes all signal preroutes from the floorplan. Use this command after importing the design.

Parameters

None

`-help`	Outputs a brief description of the `delete_all_signal_preroutes` command. For a detailed description of the command, use the man command: `man delete_all_signal_preroutes`

Related Information

- Floorplanning the Design chapter in the *Innovus Stylus Common UI User Guide*

delete_bump_grid

```
delete_bump_grid
[-help]
[-grid_names string | -all ]
```

Removes the bump grid from the design. Use this command anytime after creating a bump grid.

Parameters

`-all`	Deletes all the bump grids. *Data_type*: bool, optional
`-grid_names` *string*	Specifies the name(s) of the bump grid to be deleted. *Data_type*: string, optional
`-help`	Outputs a brief description that includes the type and default information for each `delete_bump_grid` parameter. For a detailed description of the command and all of its parameters, use the man command: `man delete_bump_grid`

Example

- The following command deletes all the bump grids in the current design:
  ```
  delete_bump_grid -all
  ```
- The following command deletes the `myGridFront_9x9` named bump grid:
  ```
  delete_bump_grid -grid_names {myGridFront_9x9}
  ```
- The following command deletes multiple bump girds:
  ```
  delete_bump_grid -grid_names {myGridFront_9x9 myGridFront_25x25}
  ```

Related Information

- `create_bump_grid`
- Floorplanning the Design chapter in the *Innovus Stylus Common UI User Guide*

delete_exclusive_groups

```
delete_exclusive_groups
[-help]
[-all | group_name]
```

Deletes existing exclusive regions for groups. You can use this command after creating exclusive groups. It restores all attributes of exclusive groups

Parameters

`-all`	Deletes all exclusive regions. When specified, the tool resets all existing exclusive groups attributes to their default values. *Data_type*: bool, optional
group_name	Specifies the name of the group to delete. The tool deletes exclusive regions of the specified groups and all other exclusive groups that have been created together. You can specify one or multiple groups after the command. *Data_type*: string, optional
`-help`	Outputs a brief description that includes the type and default information for each `delete_exclusive_groups` parameter. For a detailed description of the command and all of its parameters, use the man command: `man delete_exclusive_groups`

Examples

- The following command deletes all exclusive regions of groups (A,B,C,D,E,F).
  ```
  delete_exclusive_groups -all
  ```
- The following command deletes exclusive regions of groups A and F.
  ```
  delete_exclusive_groups {A F}
  ```
- The following command deletes exclusive regions of groups (A,B,C).
  ```
  delete_exclusive_groups B
  ```

Related Information

- `create_group`
- `update_group`
- `create_exclusive_groups`
- Floorplanning the Design chapter in the *Innovus User Guide*

delete_io_fillers

```
delete_io_fillers
[-help]
[-cell {name_list}]
[-from coord]
[-prefix prefix]
[-ring_number num]
[-side {top | bottom | left | right}]
[-to coord]
[-logic [-derive_connectivity]]
```

Deletes I/O filler cell instances from the design. Use this command after filler cells have been added using the `add_io_fillers` command.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `delete_io_fillers` parameter. For a detailed description of the command and all of its parameters, use the man command: `man delete_io_fillers`
`-cell {name_list}`	Specifies the name list of the I/O filler cells to be removed. *Data_type*: list, optional
`-derive_connectivity`	Deletes connectivity for I/O fillers. *Data_type*: bool, optional
`-from coord`	Specifies the coordinates of the starting location from where I/O filler cells are to be removed. The values are in micrometers. *Data_type*: float, optional
`-logic`	Deletes non-physical logical I/O filler cells created by the `add_io_fillers` command. *Data_type*: bool, optional
`-prefix prefix`	Specifies the prefix of the names of I/O filler instances to be removed. *Data_type*: string, optional
`-ring_number num`	Specifies the number of the ring from which I/O filler cells are to be removed. Use this parameter if you want to delete I/O filler cell instances from only a specific I/O ring in a multiple I/O ring design. *Data_type*: int, optional
`-side {top | bottom | left | right}`	
	Specifies the side as top side, bottom side, left side, and right side of the I/O box from which the I/O filler cells are to be removed. *Default*: If you do not specify this parameter, all sides are specified. *Data_type*: string, optional
`-to coord`	Specifies the coordinates of the ending location till which I/O filler cells are to be removed. The values are in micrometers. *Data_type*: float, optional

Related Information

- `add_io_fillers`
- Floorplanning the Design chapter in the *Innovus Stylus Common UI User Guide*

delete_io_insts

```
delete_io_insts
[-help]
-insts {name1 name2 ...}
```

Deletes the specified I/O instance. You can use this command at any point in the design flow.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `delete_io_insts` parameter. For a detailed description of the command and all of its parameters, use the man command: `man delete_io_insts`
`-insts {name1 name2 ...}`	
	Specifies the name(s) of I/O instance(s) to be deleted. *Data_type*: list, required

Example

- The following command deletes the specified I/O instance:
  `delete_io_insts -insts` newIo

Related Information

- `add_io_insts`
- Floorplanning the Design chapter in the *Innovus Stylus Common UI User Guide*

delete_io_row_fillers

```
delete_io_row_fillers
[-help]
-cell {filler_cell_name_list ...}
[-logic]
[-prefix prefix]
[-io_row name]
[-from coord]
[-to coord]
```

Deletes the I/O row filler cells. You can use this command at any point in the design flow.

Parameters

`-cell {filler_cell_name_list...}`	
	Specifies the name of the filler cell(s) to delete from the I/O row. *Data_type*: string, optional
`-from coord`	Specifies the coordinates of a specified starting range where I/O filler cells are to be deleted. The values are in micrometers. *Data_type*: float, optional
`-help`	Outputs a brief description that includes the type and default information for each `delete_io_row_fillers` parameter. For a detailed description of the command and all of its parameters, use the man command: `man delete_io_row_fillers`
`-io_row name`	Specifies the name of the I/O row to delete the filler cell. *Data_type*: string, optional
`-logic`	Deletes non-physical I/O fillers. *Data_type*: bool, optional
`-prefix prefix`	Specifies the prefix name to be appended to the I/O row filler cell to be deleted. *Data_type*: string, optional
`-to coord`	Specifies the coordinates of a specified ending range where I/O row filler cells to be deleted. The values are in micrometers. *Data_type*: float, optional

Example

- The following command deletes the specified I/O instance.
  ```
  delete_io_row_fillers -cell PAD8 -prefix PAD8
  ```

Related Information

- `add_io_row_fillers`
- Floorplanning the Design chapter in the *Innovus Stylus Common UI User Guide*
  - Performing I/O Row Based Pad Placement

delete_pg_pins

```
delete_pg_pins
[-help]
[-area {x1 y1 x2 y2}]
[-layers layer_id | {list_of_layer}]
[-net net_name]
[-pg_port]
[-all | -selected ]
```

Deletes power/ground pins as per the specified parameters and reports the number of deleted power/ground pins. You can use this command at any point in the design flow.

Note: The `delete_pg_pins` command only deletes the pin shapes and fterms created using the `create_pg_pin` command. If the power/ground pins are added using the read_floorplan command, then the `delete_pg_pins` command only deletes the pin shapes and keeps the pg pins.

If there is no power/ground pin and pin box definition in the .fp.gz file, and you add the power/ground pins using the `create_pg_pin` command, then Innovus will automatically add the pg fterm for it. In such a case, the `delete_pg_pins` command deletes both pin shapes and pg pins.

Parameters

`-all`	Deletes all the power/ground pins. *Data_type*: bool, optional
`-area x1 y1 x2 y2`	
	Specifies a bounding box for the area in which power/ground pins are to be deleted. *Data_type*: rect, optional
`-help`	Outputs a brief description that includes the type and default information for each `delete_pg_pins` parameter. For a detailed description of use the man command: `man delete_pg_pins`
`-layers layer_id | {list_of_layer}`	
	Specifies the layer(s) on which the power/ground pins are to be deleted. **Note**: Layers can be specified using the LEF layer names or layer ID numbers. *Data_type*: string, optional
`-net net_name`	Specifies the name of the net whose attached power/ground pins are to be deleted. *Data_type*: (net)*, optional
`-pg_port`	Deletes pg ports with pg shape or deletes pg ports based on the net name specified with the `-net` option. *Data_type*: bool, optional
`-selected`	Deletes the selected power/ground pins. *Data_type*: bool, optional

Related Information

- `create_pg_pin`
- `read_floorplan`
- Floorplanning the Design chapter in the *Innovus Stylus Common UI User Guide*
- Floorplan Menu chapter in the *Innovus Stylus Common UI Menu Reference*

delete_place_halo

```
delete_place_halo
[-help]
[-block | -all_macros | -all_black_boxes | -all_commit_partitions  | -all_blocks | -all_io_pads | -cell name]
```

Removes block halo values for specific blocks or for all blocks. Use this command after running `create_place_halo`.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `delete_place_halo` parameter. For a detailed description of the command and all of its parameters, use the man command: `man delete_place_halo`
`-all_black_boxes`	Specifies the removal of halo values for all black boxes. *Data_type*: bool, optional
`-all_blocks`	Specifies the removal of halo values for all blocks. *Data_type*: bool, optional
`-all_commit_partitions`	Specifies the removal of halo values for all committed partitions. *Data_type*: bool, optional
`-all_io_pads`	Specifies the removal of halo values for all IO pads. *Data_type*: bool, optional
`-all_macros`	Specifies the removal of halo values for all hard macros. *Data_type*: bool, optional
`-cell name`	Specifies the removal of halo values for all instances of a cell. *Data_type*: string, optional
`-block`	Specifies the block where the halo is to be deleted. You can use wildcards (* or ?) for specifying instance names. *Data_type*: string, optional

Example

The following command removes block halo values from a hard macro named cube32:

```
delete_place_halo cube32
```

Related Information

- `create_place_halo`
- Floorplanning the Design chapter in the *Innovus Stylus Common UI User Guide*
- Floorplan Menu chapter in the *Innovus Stylus Common UI Menu Reference*
  - Edit Halo

delete_relative_floorplan

```
delete_relative_floorplan
[-help]
{obj_name_list | -all}
```

Removes the relative floorplan information from the database. It deletes all relative floorplan constraints, all relative constraints for the specified object(s), or removes the last executed relative floorplan action from the database. Use this command after running `create_relative_floorplan`.

Notes:

- The arrows showing relative constraints for the specified objects are removed from the floorplan display.
- The `delete_relative_floorplan` command accepts an array of objects.

Parameters

`-all`	Delete all relative floorplan constraints. *Data_type*: bool, optional
`-help`	Outputs a brief description that includes type and default information for each `delete_relative_floorplan` parameter. For a detailed description of the command and all of its parameters, use the man command: `man delete_relative_floorplan`
obj_name_list	Deletes all relative constraints for the specified object(s). **Note**: If you do not specify *obj_name_list*, this command removes the last executed relative floorplan action from the database. *Data_type*: string, optional

Example

- The following command deletes obj1, obj2, and obj3 from the database:
  ```
  delete_relative_floorplan {obj1 obj2 obj3}
  ```

Related Information

- `create_relative_floorplan`
- Floorplanning the Design chapter in the *Innovus Stylus Common UI User Guide*

delete_route_blockages

```
delete_route_blockages
[-help]
[-type {all routes fills slots}]
[-layers <layer+> | -all {cut route drc_region trim_metal}]
```

Deletes a routing blockage object. Use this command during partition floorplanning.

Note: The `delete_route_blockages` command matches the layer name and the layer type while deleting routing blockages. If the layer name and the layer type do not match, the specified layer is not deleted and a warning about the mismatch is displayed. This rule is honored by standard layers, cut layers, drc region layers, and trim metal layers. When a list of layers is specified, only the layers matching with the specified layer type are deleted.

Parameters

`-all {cut route drc_region trim_metal}`	
	Provides a quick way to specify layer types if all blockages on that layer type are to be deleted. You can specify multiple layer types with this parameter. • `cut`: Specify to delete blockages on all cut layers. • `route`: Specify to delete blockages on all routing layers. • `drc_region`: Specify to delete routing blockages on drc region layers. • `trim_metal`: Specify to delete routing blockages on all trim metal layers. *Data_type*: (enum)+, optional
`-help`	Outputs a brief description that includes the type and default information for each `delete_route_blockages` parameter. For a detailed description of the command and all of its parameters, use the man command: `man delete_route_blockages`
`-layers <layer+>`	Specifies the layer, list of layers, or all layers on which the routing blockage is to be deleted. *Data_type*: (layer)+, optional
`-type {all routes fills slots}`	
	Specifies the type of routing blockage to delete. The following options are available: • `all`: Deletes all types of blockages. • `routes`: Deletes the blockages that block routing. • `fills`: Deletes the blockages that block metal-fill. • `slots`: Deletes the blockages that block slots. *Default*: `all` *Data_type*: enum, optional

Examples

- The following commands create a blockage on all metal layers and deletes only on metal3:

```
create_route_blockage -rects 11.2 164.07 95.005 319.205 -all
delete_route_blockages -type all -layers 3
```

Related Information

- `create_route_blockage`
- `set_route_blockage_default_layer`
- Floorplanning the Design chapter in the *Innovus Stylus Common UI User Guide*

delete_route_halos

```
delete_route_halos
[-help]
[-litho_halo]
{-all_blocks | -block block_name_list | -cells cell_name_list | -insts instance_name | -design_halo}
```

Deletes a routing halo for a blackbox, hard macro, or block-level design. Use this command after specifying a routing halo for a blackbox, hard macro, or block-level design. You can use the `create_route_halo` command to add routing halos.

Parameters

`-all_blocks`	Deletes the routing halo for all blocks and black boxes. *Data_type*: bool, required
`-block block_name_list`	Deletes the routing halo for the specified blocks. *Data_type*: list, required
`-cells cell_name_list`	Deletes the routing halo for the specified cells. *Data_type*: list, required
`-design_halo`	Deletes the routing halo at the design level. *Data_type*: bool, required
`-help`	Outputs a brief description that includes the type and default information for each `delete_route_halos` parameter. For a detailed description of the command and all of its parameters, use the man command: `man delete_route_halos`
`-insts instance_name`	Specifies that the routing halo should be added to the specified instance. Use this option to delete the halo of an I/O pad instance. *Data_type*: list, required
`-litho_halo`	Deletes the litho routing halo. *Data_type*: bool, optional

Example

```
delete_route_halos -all_blocks
```

Related Information

- `create_route_halo`
- Floorplanning the Design chapter in the *Innovus Stylus Common UI User Guide*
- Floorplan Menu chapter in the *Innovus Stylus Common UI Menu Reference*
  - Edit Halo

delete_row

```
delete_row
[-help]
{-all | -selected | -site string| -row}
```

Deletes the specified row(s). Use this command after importing a design.

Parameters

`-all`	Specifies that all rows will be deleted. *Data_type*: bool, required
`-help`	Outputs a brief description that includes the type and default information for each `delete_row` parameter. For a detailed description of the command and all of its parameters, use the man command: `man delete_row`
-row	Specifies the name or id of the row(s) to be deleted. *Data_type*: list, required
`-selected`	Specifies that the selected rows should be deleted. *Data_type*: bool, required
`-site` *string*	Specifies the site for which rows will be deleted. *Data_type*: string, required

Example

The following command deletes rows for the site site0A10.

```
delete_row -site site0A10
```

Related Information

- `create_row`
- `init_core_rows`
- `split_row`
- `stretch_rows`
- Floorplanning the Design chapter in the *Innovus Stylus Common UI User Guide*
  - Creating and Editing Rows

delete_selected_from_floorplan

```
delete_selected_from_floorplan
[-help]
```

Deletes the currently selected floorplan objects. You can use this command at any point in the design flow.

Parameters

None

`-help`	Outputs a brief description of the `delete_selected_from_floorplan` command. For a detailed description of the command, use the man command: `man delete_selected_from_floorplan`

Related Information

- `delete_all_floorplan_objs`
- Floorplanning the Design chapter in the *Innovus Stylus Common UI User Guide*
- Floorplan Menu chapter in the *Innovus Stylus Common UI Menu Reference*
  - Clear Floorplan

finish_floorplan

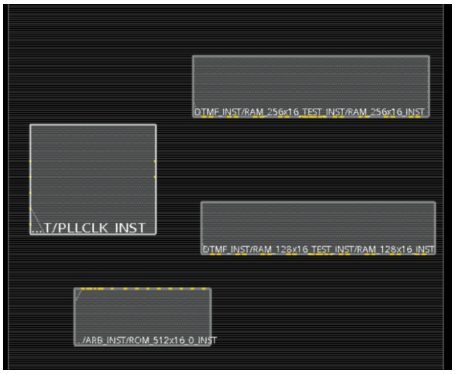
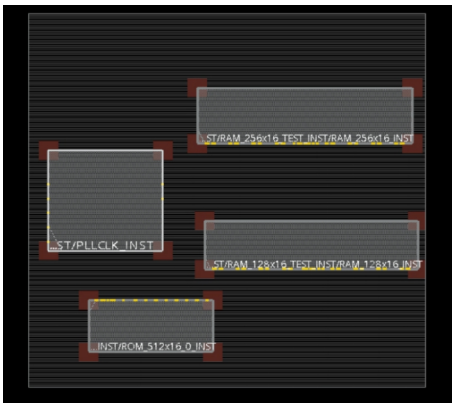
```
finish_floorplan
[-help]
[-mask integer]
{ [-auto_halo | [-auto_halo_based_on_pitch [-margin value]] | -add_halo width |
[-fill_place_blockage blockage_type max_gap
   [-density value] [-exclude_flops ][-dead_area float] [-honor_placement_density ]] |
[-fill_route_blockage max_gap [-layers layer_id]
[-cut_layers cut_layer_id]
[-fills ] [-pg_net_exception ]
[-spacing space_value | -design_rule_width width][-dead_area max_dead_area]] |
-add_corner_blockage {x y} | [-drc_region_layer layer+ [-edge_extend {x y}] [-edge_shrink {x y}]] |-add_boundary_end_of_line_blockage | -
add_row_based_region_blockage | -undo ]
[-area {x1 y1 x2 y2}]
[-name_prefix name] }
```

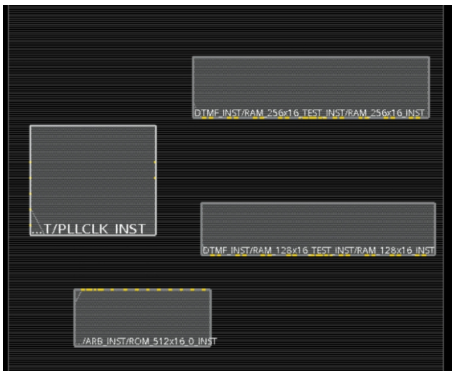
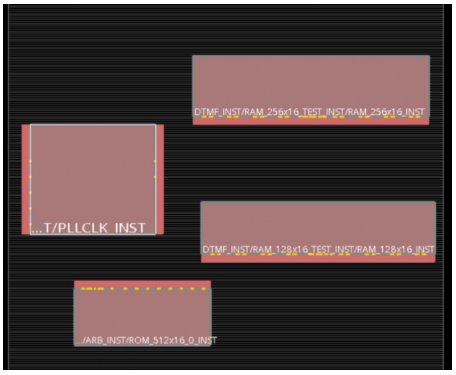
Performs advanced placement-related refinements to a floorplan, to produce a more polished floorplan. The `finish_floorplan` command, can be used on any floorplan, including third-party generated floorplans. You can use the `finish_floorplan` Category Attributes to set the active objects and specify the channel direction for the `finish_floorplan` command to use. The `finish_floorplan` Category Attributes affect the behavior of the `finish_floorplan` command.

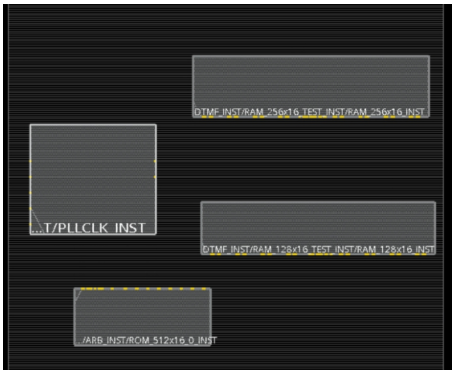
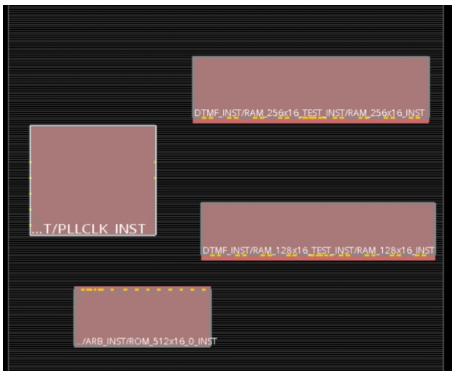
Note: The `finish_floorplan` command honors the settings of the `floorplan_no_cut_row` attribute that specifies if rows are not required to be cut before other operations. When the `floorplan_no_cut_row` attribute is disabled, the `finish_floorplan` command adds blockages around the current row/non-row area in the design when the `finish_floorplan_active_objs row` or `finish_floorplan_drc_region_objs nonRowArea` options are set. If rows are not cut before other operations but the `floorplan_no_cut_row` attribute is enabled, the `finish_floorplan` command can work as cut row flow to add blockages around row/non-row area which satisfy row constraints.

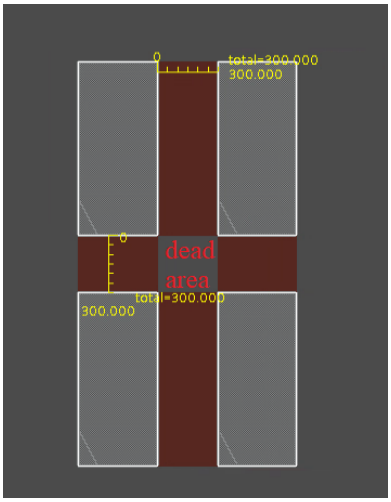
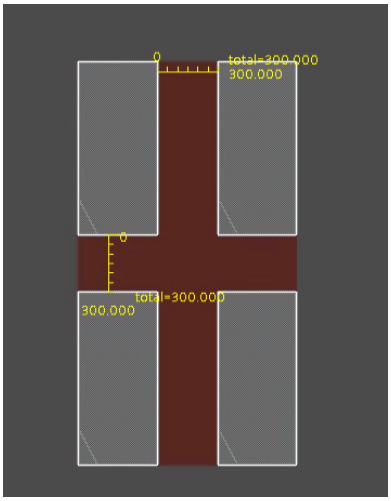
Parameters

`-add_boundary_end_of_line_blockage`	Adds end-of-line(EOL) routing blockage with “COREEOLBLOCKAGE” property around active object boundary. This parameter can only work with rows as active objects. For each layer, EOL routing blockage is perpendicular with prefer routing direction.
	Note: A Core EOL blockage rule is used to specify line-end blockages formed from the core edges. The EOL Blockage is defined in LEF as below:
	`PROPERTY LEF58_COREEOLBLOCKAGE "COREEOLBLOCKAGE outwardSize inwardSize [SIDEEXTENSION sideExtension] ;" ;`
	`COREEOLBLOCKAGE outwardSize inwardSize SIDEEXTENSION sideExtension`
	Defines the line-end blockages formed from the core edges in the orthogonal direction of the preferred routing direction of the layer of a design. The blockage is formed by going outward from the core by outwardSize and inward by inwardSize. If SIDEEXTENSION is defined, the blockage is extended by sizeExtension along the direction of the core edge on both sides. These blockages prevent any line-end of a wire end inside it. If a wire goes through the blockage and just touches the far end edge, it is still a violation. If a wire just touches an edge without extending inside the blockage, it is fine. .
	Data_type: bool, optional

`-add_corner_blockage {x y}`	Specifies the width and height of the hard placement blockage that is automatically added around the corner. Example: `finish_floorplan -add_corner_blockage 50 50` Before：  After：  *Data_type*: string, optional
`-add_halo width`	Adds halo of the specified width. *Data_type*: float, optional
`-add_row_based_region_blockage`	When specified, it adds row based region blockages. This parameter adds region layer blockages with “ROWBASED” property in some row areas. **Note:** The row area is determined by standard cells with both such row sites and OBS on the region layer. *Data_type*: bool, optional
`-area {x1 y1 x2 y2}`	Specifies the lower left x coordinate, lower left y coordinate, upper right x coordinate, and upper right y coordinate respectively of the blockage area. *Data_type*: rect, optional

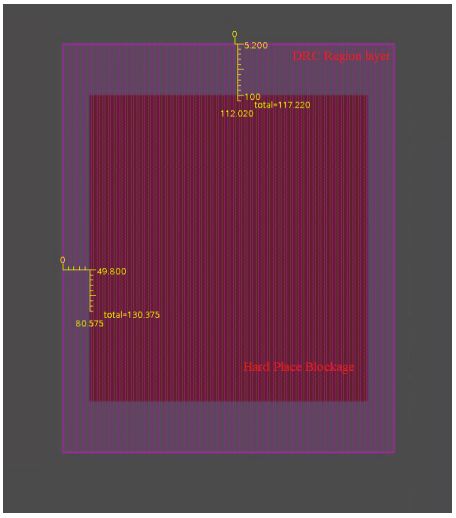
`-auto_halo`	Automatically add halos around every macro in design. For each macro edge, the halo thickness is proportional to pin density on that edge. Example: `finish_floorplan -auto_halo` Before:  After:  *Data_type*: bool, optional

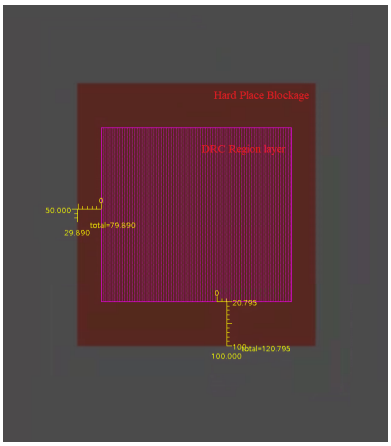
`-auto_halo_based_on_pitch`	Automatically add halos based on the number of pins, routing layer, and layer pitch on the edge. Example: `finish_floorplan -auto_halo_based_on_pitch` Before:  After:  *Data_type*: bool, optional
`-cut_layers` *cut_layer_id*	Specifies the cut layer ID on which the routing blockage is to be applied. *Data_type*: string, optional

`-dead_area` *float*	Specifies the maximum value of dead area.
	Note: A dead area is an island with placeable rows that has a narrow channel connect to main core area. For example, the placeable rows surrounded by macros.
	Example：
	`finish_floorplan -fill_place_blockage hard 300`
	 
	
	`finish_floorplan -fill_place_blockage hard 300 -dead_area 90000`
	 
	
	Data_type: float, optional

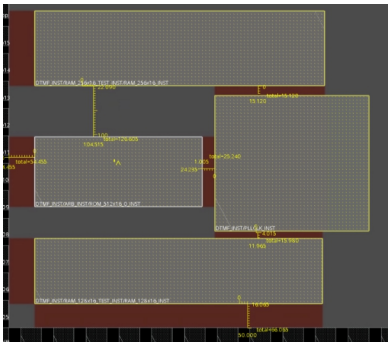
`-density` *value*	Specifies the density of the partial placement blockage. *Data_type*: string, optional

`-design_rule_width width`	Specifies the effective width of the routing blockage. The value is a real number greater than 0. *Data_type*: float, optional
`-drc_region_layer layer+`	Specifies the region layer name list. *Data_type*: string, optional

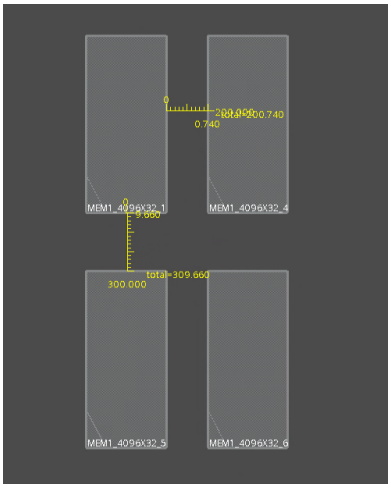
`-edge_extend {x y}`	Specifies that the drc region can be extended from the specified edge. The `-edge_extend` parameter is used for creating drc region blockages by extending from the specified drcRegion objects.
	Note:
	• The specified value for option `-edge_extend` should be equal to or greater than zero. If the specified value is less than zero, treat it as zero.
	• For X/Y direction, if the `-edge_extend` and `-edge_shrink` options are specified at the same time, then Innovus compares the value first and then extends or shrinks. Example, `-edge_extend {3 4}` and `-edge_shrink {2 5}` are specified at the same time, tool will extend 1(3-2) in X-direction and Shrink 1(5-4) in Y-direction.
	Example:
	`set_db finish_floorplan_drc_region_objs hard_blockage`
	`finish_floorplan -drc_region_layer R1 -edge_extend {50 100}`
	 
	
	Data_type: point, optional

`-edge_shrink {x y}`	Specifies that the drc region can be shrunk from the specified edge. The `-edge_shrink` parameter is used for creating drc region blockages by shrinking the specified drcRegion objects.
	Note:
	• The specified value for option `-edge_shrink` should be equal to or greater than zero. If the specified value is less than zero, treat it as zero.
	• For X/Y direction, if the `-edge_extend` and `-edge_shrink` options are specified at the same time, then Innovus compares the value first and then extends or shrinks. Example, `-edge_extend {3 4}` and `-edge_shrink {2 5}` are specified at the same time, tool will extend 1(3-2) in X-direction and Shrink 1(5-4) in Y-direction.
	Example:
	`set_db finish_floorplan_drc_region_objs hard_blockage`
	`finish_floorplan -drc_region_layer R1 -edge_shrink {50 100}`
	 
	
	Data_type: point, optional

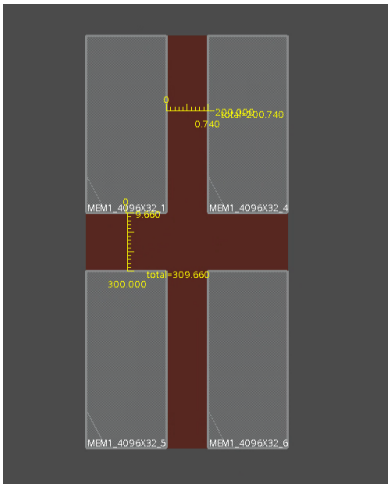
`-exclude_flops`	Specifies that flops and latches will be excluded from the partial placement blockage. **Note**: The `-density` and `-exclude_flops` options must be used with the `-fill_place_blockage` parameter. *Data_type*: bool, optional
`-fill_place_blockage blockage_type max_gap`	

	Adds placement blockages of the specified blockage type (*`blockage_type`*) to fill macro-to-macro, macro-to-boundary, macro-to-OBS, OBS-to-OBS, and boundary-to-OBS gaps up to a specified size (*`max_gap`*).
	For *`blockage_type`*, you can specify soft, hard, or partial.
	For *`max_gap`*, specify a positive value, in microns.
	Example1:
	`finish_floorplan -fill_place_blockage hard 50`
	 
	Example2:
	`set_db finish_floorplan_active_objs {macro hard_blockage}`
	`finish_floorplan -fill_place_blockage hard 300`

Before:

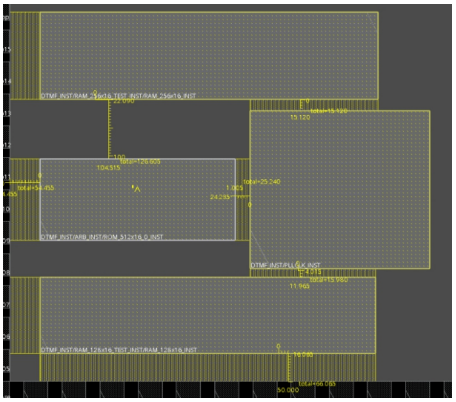


After:



Data_type: string, optional

`-fill_route_blockage` *max_gap*

	Adds routing blockage upto a specified size (*max_gap*) Example: `finish_floorplan -fill_route_blockage 50`  *Data_type*: float, optional
`-fills`	Specifies that the routing blockage is to be applied on metalfills. *Data_type*: bool, optional
`-help`	Outputs a brief description that includes type and default information for each `finish_floorplan` parameter. For a detailed description of the command and all of its parameters, use the man command: `man finish_floorplan`
`-honor_placement_density`	Adds standard cell density check when adding placement blockages. **Note:** • If option the `-density` parameter is not specified with the "`finish_floorplan -fill_place_blockage soft`" command, the density of the created soft blockage is 100. • If there are standard cells in the channel, partial blockages will be created with the density of the current standard cell in the channel when the following command is used: `finish_floorplan -fill_place_blockage hard/soft -honor_placement_density` • If there are standard cells in the channel and the specified density for the partial blockage is less than the density of the current standard cell in the channel, partial blockages will be created with the density of the current standard cell in the channel with the "`finish_floorplan -fill_place_blockage partial -density`" command. • If there are standard cells in the channel and the specified density for partial blockage is larger than the density of the current standard cell in the channel, partial blockages will be created with the specified density when the "`finish_floorplan -fill_place_blockage partial -density`" command is called. *Data_type*: bool, optional
`-layers` *layer_id*	Specifies the layer ID of the route blockage. *Data_type*: string, optional
`-margin` *value*	Specifies the pitch margin when halos are added using `-auto_halo_based_on_pitch`. *Default*: `0.1` *Data_type*: float, optional

`-mask integer`	Specifies the mask number used when adding routing blockages and creating DRC region.
	• The `-mask` parameter enables the `finish_floorplan` command to block routing layers with specified masks when adding routing blockages and creating DRC region. Layers with other masks can still be routed through the blockage.
	• The `-mask` parameter must be used with one or more of `-layers`, `-cut_layers` and `-drc_region_layer` parameters.
	• Only one mask can be specified. The default value of the `-mask` parameter is 0. This implies that all the specified layers will be blocked. Uncolored routes are always blocked.
	• If the `-mask` parameter is specified with an integer value greater than 0, then both layers with specified mask and uncolored layers are blocked.
	Example. If `-mask 1` is specified, it blocks mask 1 and uncolored routes.
	• If more than one layers are associated with a mask, the mask is applied to all of them.
	• If the `-mask` parameter is specified, the tool checks if the mask is suitable for all of the specified layers. If not, it displays an ERROR message and no blockages are created. The number of arguments specified with the –`mask` parameter must be same as the number of arguments specified with the `-layers`, `-cut_layers` and `-drc_region_layer` parameters.
	• The NanoRoute router and the `check_drc` command honor the `-mask` parameter.
	Example:
	• The following command creates routing blockages for blocking mask 2 of M1 layer.
	`finish_floorplan -layers M1 -mask 2 -fill_route_blockage 400`
	• The following command(s) DRC region for mask 1 of PIO layer in core to die area.
	`set_db finish_floorplan_drc_region_objs {core_spacing}`
	`finish_floorplan -drc_region_layer PIO -mask 1 -edge_extend {0.083 0}`
	Default: `0`
	Data_type: int, optional

`-name_prefix name`	Specifies the name prefix for the added blockage to avoid naming conflict with the existing one. *Data_type*: string, optional
`-pg_net_exception`	Specifies that the routing blockage is to be applied on a signal net routing and not on power or ground net routing. *Data_type*: bool, optional
`-spacing space_value`	Specifies the minimum spacing between layers within the routing blockage. The value is a real number greater than or equal to 0. *Data_type*: float, optional
`-undo`	Reverses the previous run of `finish_floorplan`. *Data_type*: bool, optional

Example

- The following commands specify the region layer name list for creating a region layer for the specified DRC region object:
  ```
  set_db finish_floorplan_drc_region_objs macro
  finish_floorplan -drc_region_layer R1
  ```
- The following command creates a dead area with partial placement blockage of gap of 5 microns and density 60:
  ```
  finish_floorplan -dead_area 100 -fill_place_blockage partial 5 -density 60
  ```

Related Information

- `finish_floorplan` Category Attributes
- Floorplanning the Design chapter in the *Innovus Stylus Common UI User Guide*

finish_floorplan Category Attributes

The root attributes in the `finish_floorplan` category are:

- finish_floorplan_abutted_edge_as_non_row_area
- finish_floorplan_active_objs
- finish_floorplan_add_blockage_direction
- finish_floorplan_drc_region_objs
- finish_floorplan_override

To set the value of any root attribute, use the `set_db` command:

```
set_db attribute_name value
```

To get the current value of any root attribute, use the `get_db` command:

```
get_db attribute_name
```

To get the current value of all attributes in the `finish_floorplan` category, use the following command:

```
get_db -category finish_floorplan
```

finish_floorplan_abutted_edge_as_non_row_area

`finish_floorplan_abutted_edge_as_non_row_area {true|false}`

Default: `true`

Data_type: bool, read/write

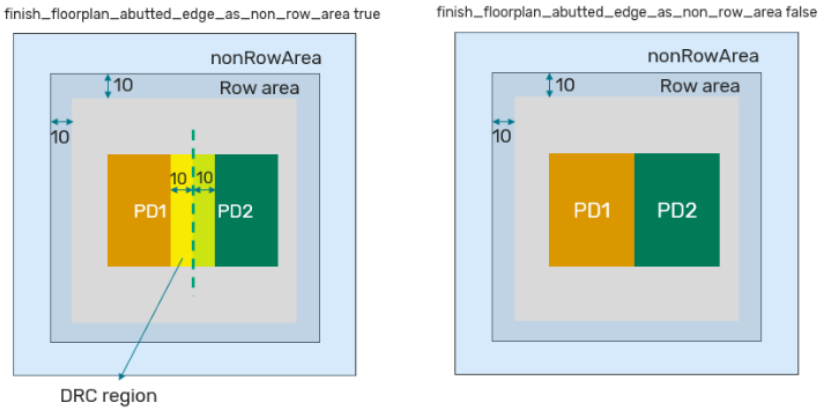
Specifies whether the abutted edge of a power domain should treated be as nonRowArea.

- `true`: The abutted edge of the power domain is treated as nonRowArea with size 0 by default. The tool creates a DRC region around the abutted edge with the `finish_floorplan -edge_extend` parameter option.
- `false`: The abutted edge of the power domain is excluded from nonRowArea. The tool then does not create a DRC region around the abutted edge with the `finish_floorplan -edge_extend` parameter option.

Note: This attribute can only be specified when the `finish_floorplan_drc_region_objs non_row_area` attribute is also specified.

For example:

```
set_db finish_floorplan_drc_region_objs non_row_area
set_db finish_floorplan_abutted_edge_as_non_row_area {true|false}
finish_floorplan -drc_region_layer M1 -edge_extend {10 10}
```



Related Commands

- `finish_floorplan`

finish_floorplan_active_objs

`finish_floorplan_active_objs obj_list`

Default: `{macro macro_halo core}`

Data_type: enum* {macro macro_halo core io_pad io_cell fence hard_blockage soft_blockage partial_blockage route_blockage row}, read/write

Specifies the active object list. The boundaries of the specified active objects are looked at while deciding where to add blockages. Non-active objects are treated like they do not exist by the `finish_floorplan` command.

Active objects can be:

- `core`
- `macro`
- `macro_halo`
- `io_cell`
- `io_pad`
- `fence`
- `hard_blockage`
- `soft_blockage`
- `partial_blockage`
- `route_blockage`
- `routing_blockage`
- `row`

Related Commands

- `finish_floorplan`

finish_floorplan_add_blockage_direction

`finish_floorplan_add_blockage_direction {x | y | xy}`

Default: `xy`

Data_type: enum {x y xy}, read/write

Specifies the direction in which the blockage is added. You specify one of the following options:

- *x*: Fills the channel whose height is less than width.
- *y*: Fills the channel whose width is less than height.
- *xy*: Fills every channel whose smallest size (in X or Y direction) is within Max Gap range.

Related Commands

- `finish_floorplan`

finish_floorplan_drc_region_objs

`finish_floorplan_drc_region_objs obj_list`

Default: `macro macro_halo hard_blockage min_gap core_spacing`

Data_type: enum* {macro macro_halo hard_blockage min_gap core_spacing non_row_area}, read/write

Specifies the drc region object list. The boundaries of the specified drc region objects are looked at while deciding where to add blockages.

The drc region objects can be:

- `core_spacing`

- hard_blockage
- macro
- macro_halo
- min_gap
- non_row_area

Related Commands

- finish_floorplan

finish_floorplan_override

```
finish_floorplan_override {true | false}
```

Default: 0

Data_type: bool, read/write

Overrides old blockages that overlap with the new blockages.

Related Commands

- finish_floorplan

fix_boundary_overlaps

```
fix_boundary_overlaps
[-help]
[-move_area {x1 y1 x2 y2}]
[-groups {string1 string2 ...}]
[-hinsts {string1 string2 ...}]
[-fix_area {x1 y1 x2 y2}]
```

Resolves the overlaps between macros or hierarchical instances (hinsts). This command can also be used to resolve overlaps between instance groups.

Parameters

`-fix_area {x1 y1 x2 y2}`	
	Specifies the coordinates of the area in which overlap needs to be resolved. *Data_type*: rect, optional
`-groups {string1 string2 ...}`	
	Specifies the prefixed group names. Use this parameter if you want to resolve overlaps between only specified groups. *Data_type*: list, optional
`-help`	Prints a brief description that includes type and default information for each `fix_boundary_overlaps` parameter. For a detailed description of the command and all of its parameters, use the man command: `man fix_boundary_overlaps`
`-hinsts {string1 string2 ...}`	
	Specifies the prefixed hinst names. Use this parameter if you want to resolve overlaps between only specified hinsts. *Data_type*: list, optional
`-move_area {x1 y1 x2 y2}`	
	Specifies the area where the floorplan objects can be moved when resolving overlaps. *Data_type*: rect, optional

Related Information

- Floorplanning the Design chapter in the *Innovus Stylus Common UI User Guide*

fix_floorplan

```
fix_floorplan
[-help]
-file fileName
-type {ufc tcic}
[-selected ]
[-report_file string]
[-report_unplaced_block ]
[-break_at_conflict ]
[-honor_snap_block_grid ]
[-selected_rules rule_name_list]
```

Fixes the detected violations as part of the UFC methodology by adding routing blockages around macro, snapping macro location, or re-shaping core shape. The UFC methodology also provides the capability to automatically fix the violations with imported UFC rules.

Note: The Unified Floorplan Constraints (UFC) methodology is a powerful Innovus capability that can define several rules on floorplan objects, check specified rules, and report the violations with rule name as well as severity. The violations can be shown in the violation browser. UFC also provides the ability to automatically fix specific violations. It can add routing blockages, snap macro location, adjust core shape to fix corresponding violations. In addition, UFC also provides the ability to exclude rules from checking for specified objects.

Usage Model:

- Load the Design.
- Create a UFC file with floorplan rules.
- Run UFC checking. (check_ufc)
- Fix reported violations. (fix_floorplan)

You can use the `fix_floorplan` command to fix the detected violations as part of the UFC methodology. It fixes violations by:

- Adding routing blockages around macros
- Snapping macro locations
- Re-shaping core shapes

The UFC methodology also provides the capability to automatically fix the violations with imported UFC rules.

Note: The `fix_floorplan` command tries to fix all the detected floorplan violations by default, however, it currently does not support fixing die size and core size violations. If a floorplan has rule conflicts, then the results after the fix may not be perfect. Cadence recommends you ensure that the rules used are not conflicting and that the die or core size satisfy the rules. Cadence also recommends you contact your Cadence representative if your constraints file comes from a third party, but you want to use the `fix_floorplan` command to fix the violations.

Note: The `fix_floorplan` command honors the `floorplan_no_cut_row` attribute that specifies if rows are not required to be cut before operations. With `floorplan_no_cut_row false` or not specified, the `fix_floorplan` command will work based on the current row shape. If there is any violation for rows, the `fix_floorplan` command will run the `split_row` command to fix. With `floorplan_no_cut_row true`, the tool identifies that the rows should not be cut. The `fix_floorplan` command will fix the row violations by adding place blockages instead of cutting row. The place blockages will be added to the row area that will be cut out and kept after fix_floorplan to satisfy the row constraint. The actual row shape will not be changed after the `fix_floorplan` command.

Parameters

`-break_at_conflict`	Enables the tool to exit from fixing violations when conflict of rules is encountered. You must fix the conflict of rules and then re-run the `fix_floorplan` command. *Data_type*: bool, optional
`-file` *fileName*	Specifies the name of file containing the rules or constraints. *Data_type*: string, required
`-help`	Prints a brief description that includes type and default information for each `fix_floorplan` parameter. For a detailed description of the command and all of its parameters, use the man command: `man fix_floorplan`
`-honor_snap_block_grid`	When specified, the `fix_floorplan` command honors the block snap grid defined by the `set_db floorplan_snap_block_grid` attribute. *Data_type*: bool, optional
`-report_file` *string*	Specifies the name of the file that saves the report summary. *Data_type*: string, optional
`-report_unplaced_block`	Enables the tool to show the number of unplaced macros in summary. *Data_type*: bool, optional
`-selected`	Specifies that only the selected objects can be moved to fix violations. **Note:** The `fix_floorplan` command currently only support selected macros. *Data_type*: bool, optional
`-selected_rules` *rule_name_list*	When specified, the `fix_floorplan` command only fixes violations for specified rules. The `-selected_rules` parameter accepts rule names, rule names list, variables, and wildcards. **Note:** When this option is not specified, the `fix_floorplan` command fixes all violations by default. *Data_type*: bool, optional
`-type {ufc tcic}`	Specifies the type of input file. The `fix_floorplan` command supports ufc and tcic files. *Data_type*: enum, required

Example

- The following command fixes the ufc violations based on the rules defined in the DTMF1.ufc file and writes the report to the fix_cmds.tcl file.
  ```
  fix_floorplan -type ufc -file DTMF1.ufc -report_file fix_cmds.tcl
  ```
- The following command fixes the ufc violations for Rule A and Rule B only.
  ```
  fix_floorplan -type tcic -file test.tcic -selected_rules "RuleA RuleB"
  ```

Related Information

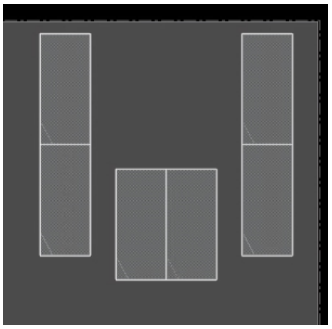
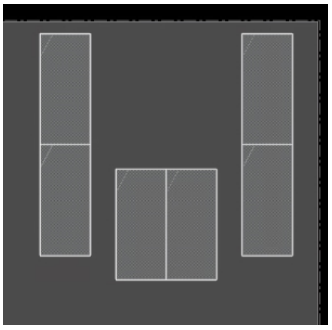
- `check_ufc`
- Floorplanning the Design chapter in the *Innovus Stylus Common UI User Guide*

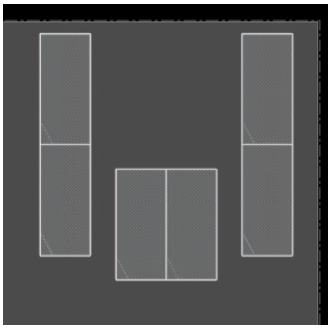
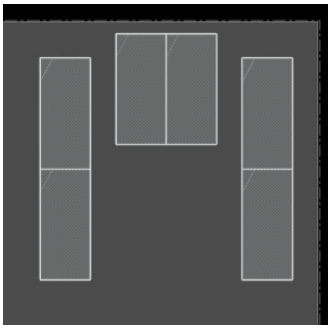
flip_or_rotate_obj

```
flip_or_rotate_obj
[-help]
[-keep_relative]
{-flip {MX | MY} | -rotate {R90 | R180 | R270}}
[-group]
[-objs object]
```

Flips or rotates the selected objects. Alternatively, you can use the "*Flip/Rotate Selected Objects*" form in the Innovus GUI, to flip or rotate an instance. Use this command after importing the design.

Parameters

`-flip {MX | MY}`	Flips the selected object(s) through x axis or y axis. Example: flip_or_rotate_obj `-flip MX` Before:  After:  *Data_type*: string, required
`-help`	Outputs a brief description that includes the type and default information for each `flip_or_rotate_obj` parameter. For a detailed description of the command and all of its parameters, use the man command: `man flip_or_rotate_obj`

`-group`	Flips or rotates the bounding box which encloses all the selected objects. Example: `flip_or_rotate_obj -flip MX -group` Before:  After:  *Data_type*: bool, optional
`-keep_relative`	Calls the `create_relative_floorplan` command. With this option enabled, when the anchor objects are rotated, the relative objects are rotated and updated automatically. *Data_type*: bool, optional
`-objs object`	Specifies the name of the object you want to flip or rotate. **Note**: The objects can be modules, groups, placement blockage, routing blockage, hard Macros, special routings, special wires, and special vias (including 45 degree). *Data_type*: string, required
`-rotate {R90|R180|R270}`	Rotates the selected object(s) by 90 or 180 or 270 degrees. *Data_type*: string, required

Example

The following command flips the selected object(s) through the x axis:

```
flip_or_rotate_obj  -flip MX
```

Related Information

- `align_selected`
- `space_selected`
- `create_relative_floorplan`
- Floorplanning the Design chapter in the *Innovus Stylus Common UI User Guide*

floorplan Category Attributes

The root attributes in the `floorplan` category are:

- floorplan_auto_sync_master_clone
- floorplan_cells_for_extra_sites
- floorplan_check_types
- floorplan_cut_off_place_blockage_outside_die
- floorplan_cut_off_route_blockage_outside_die
- floorplan_default_blockage_name_prefix
- floorplan_default_power_domain_site
- floorplan_default_tech_site
- floorplan_enable_rectilinear_design
- floorplan_extra_sites
- floorplan_force_honor_snap_rule
- floorplan_include_io_when_init_area
- floorplan_initial_all_compatible_core_site_rows
- floorplan_keep_rows_when_moving_power_domain
- floorplan_max_io_height
- floorplan_minimum_sites
- floorplan_move_macros_with_constraint
- floorplan_move_preplaced_std_cell_only
- floorplan_move_std_cell_with_constraint
- floorplan_narrow_channel_threshold
- floorplan_no_cut_row
- floorplan_finfet_inst_grid
- floorplan_finfet_manufacturing_grid
- floorplan_finfet_placement_grid
- floorplan_inst_grid
- floorplan_layer_track_grid
- floorplan_manufacturing_grid
- floorplan_placement_grid
- floorplan_power_rail_layer
- floorplan_row_site_height
- floorplan_row_site_width
- floorplan_snap_all_corners_to_grid
- floorplan_snap_block_grid
- floorplan_snap_constraint_grid
- floorplan_snap_core_grid
- floorplan_snap_die_grid
- floorplan_snap_io_grid
- floorplan_snap_place_blockage_grid
- floorplan_snap_place_blockage_type
- floorplan_user_define_grid
- floorplan_vertical_row

To set the value of any root attribute, use the `set_db` command:

```
set_db attribute_name value
```

To get the current value of any root attribute, use the `get_db` command:

```
get_db attribute_name
```

To get the current value of all attributes in the `floorplan` category, use the following command:

```
get_db -category floorplan
```

floorplan_auto_sync_master_clone

`floorplan_auto_sync_master_clone {busGuides|constraints|stdcells|macros}`

Default: `" "`

Data_type: enum* {busGuides constraints stdcells macros}, read/write

Specifies the objects that automatically sync up from master to clones during add, delete, edit, move, or stretch operations. The sync-up objects include busGuides and macros.

- `busGuides`: Any change (add, delete, edit, move, or stretch) for busGuides in master will sync-up to clones or vice versa.
- `constraints`: Any change (move, stretch, add, delete, or edit) in fence/guide/region of master will sync-up to clones or vice versa.
  - When constraints is not specified, tool will not sync-up any change of fence/guide/region between master and clones.
  - When constraints is specified, tool will auto sync-up any change of fence/guide/region between master and clones when doing moving/stretching/add/delete/edit.
- `stdcells`: Any change (add, delete, edit, move, flip, rotate, or unplace) in standard cells will sync-up to clones or vice versa.
  For example, `set_db floorplan_auto_sync_master_clone {stdcells}`
- `macros`: Any change (add, delete, edit, move, flip, rotate, or unplace) in hard macros in master will sync-up to clones or vice versa.
  For example, `set_db floorplan_auto_sync_master_clone {macros}`

Related Commands

- `finish_floorplan`

floorplan_cells_for_extra_sites

`floorplan_cells_for_extra_sites cell_name`

Default: `" "`

Data_type: string, read/write

Specifies the cell names for which the additional sites will be created. These sites can be used up later in the flow.

Related Commands

- `finish_floorplan`

floorplan_check_types

`floorplan_check_types {basic fence odd_even_site_row macro_pin color alignment_following_pin alignment_partition_clone bus_guide_connectivity feed_through partition_in_partition multi_layer_pin power_domain partition place same_length_site narrow_channel block_only unified_vdd_on_bottom all}`

Default: `basic`

Data_type: enum* {basic fence odd_even_site_row macro_pin color alignment_following_pin alignment_partition_clone bus_guide_connectivity feed_through partition_in_partition multi_layer_pin power_domain partition place same_length_site narrow_channel block_only unified_vdd_on_bottom all}, read/write

Specifies the check types for the `check_floorplan` command.

- `basic`: Checks the basic quality of the floorplan to detect potential problems before the design is passed on to other tools.
- `fence`: Checks the fences.

- `odd_even_site_row`: Checks odd or even site row. It checks if the current row fits to the setting of `setFPlanMode -rowSiteWidth` and `-rowSiteHeight`.
- `macro_pin`: Checks if the macro pins are off track.
- `color`: Checks the color issues of all floorplan objects, such as pins, preRoutes, prePlacedInst, etc.
- `alignment_following_pin`: Checks if the centerline of the followpin is the same as the row's border and matches the power attribute.
- `alignment_partition_clone`: Checks that partition clones are aligned with their master on the power mesh. This checks that the master and clones partitions are inside a core boundary, and ensures that the partition fence instances sizes match the master instance.
- `bus_guide_connectivity`: Checks incorrect overlapping and connectivity.
- `feed_through`: Checks feedthrough buffer insertion.
- `partition_in_partition`: Checks if child partitions reside inside parent partitions.
- `multi_layer_pin`: Checks multi layer pins.
- `power_domain`: Checks the power domains.
- `partition`: Checks the partitions.
- `place`: Checks the placement.
- `same_length_site`: Checks the same length site rule where, in a group of continuous poly stripes with the same length, the number of sites must be >= N. (N is specified by the `floorplan_minimum_sites` attribute.)
- `narrow_channel`: Checks for narrow channels.
- `block_only`: Checks for performance most frequently used in block implementation. Includes: basic, oddEvenSiteRow, macroPin, and color.
- `unified_vdd_on_bottom`: Checks whether all the cells in the design are with the same vdd on bottom.
- `all`: Performs all checks on the floorplan to detect potential problems before the design is passed on to other tools.

Related Commands

- `check_floorplan`

floorplan_cut_off_place_blockage_outside_die

```
floorplan_cut_off_place_blockage_outside_die {true | false}
```

Default: `true`

Data_type: bool, read/write

When creating placement blockages, it cuts the part outside of die and snaps it by default.

Related Commands

- `finish_floorplan`

floorplan_cut_off_route_blockage_outside_die

```
floorplan_cut_off_route_blockage_outside_die {true | false}
```

Default: `true`

Data_type: bool, read/write

When creating routing blockages around inst/Hinst/Partition, it cuts the part outside of die and snaps it by default. If this option is set to false, then the outside part is not cut and snapped.

Related Commands

- `finish_floorplan`

floorplan_default_blockage_name_prefix

`floorplan_default_blockage_name_prefix` *`prefix_name`*

Default: {}

Data_type: string, read/write

Names the prefix for naming placement blockages and routing blockages.

Related Commands

- `create_place_blockage`
- `create_route_blockage`

floorplan_default_power_domain_site

`floorplan_default_power_domain_site {true | false}`

Default: `false`

Data_type: bool, read/write

Create rows based on default power domain site instead of the design's default site.

Related Commands

- `create_floorplan`

floorplan_default_tech_site

`floorplan_default_tech_site` *`string`*

Default: `""`

Data_type: string, read/write

Specifies a site as default technical site. After setting a forced one, the auto-calculating default-technical-site functionality is disabled.

Note: The `write_floorplan`, `read_floorplan`, `init_core_rows` and `check_floorplan` commands honor the default technical site specified using the `floorplan_default_tech_site` attribute.

Related Commands

- `check_floorplan`
- `read_floorplan`
- `write_floorplan`

floorplan_enable_rectilinear_design

`floorplan_enable_rectilinear_design {true | false}`

Default: `false`

Data_type: bool, read/write

Specifies whether the design boundary can be changed to rectilinear shape.

Related Commands

- `check_floorplan`
- `read_floorplan`
- `write_floorplan`

floorplan_extra_sites

`floorplan_extra_sites` *`site_name`*

Default: `" "`

Data_type: string, read/write

Specifies the name for the additional sites that will be created for cells which are not used in the netlist by default. These sites can be used up later in the flow.

Note: Sites for all specified cells should be defined in the LEF.

Related Commands

- `finish_floorplan`

floorplan_force_honor_snap_rule

`floorplan_force_honor_snap_rule {true|false}`

Default: `false`

Data_type: bool, read/write

When enabled, objects are snapped to non-FinFET grids.

Note: If a design has a FinFET grid, during floorplanning, objects will automatically be snapped to the corresponding FinFET grid even if the snap grid is an instance grid, manufacture grid or a placement grid. However, with the `floorplan_force_honor_snap_rule` attribute set to `true`, the instance grid, manufacture grid, and placement grid will not be mapped to the FinFET Instance Grid, FinFET Manufacture Grid, and FinFET Placement Grid respectively.

Related Commands

- `finish_floorplan`

floorplan_include_io_when_init_area

`floorplan_include_io_when_init_area {true | false}`

Default: `false`

Data_type: bool, read/write

While calculating the target utilization (TU), if the variable is set to `true`, the module area is calculated by including the I/O cell. If set to `false`, the area of module is calculated without including the I/O cell.

Related Commands

- `create_floorplan`

floorplan_initial_all_compatible_core_site_rows

`floorplan_initial_all_compatible_core_site_rows {true | false}`

Default: `false`

Data_type: bool, read/write

When enabled, initCoreRow creates SITES which are used by all cells in the library.

Related Commands

- `create_floorplan`

floorplan_keep_rows_when_moving_power_domain

`floorplan_keep_rows_when_moving_power_domain {true | false}`

Default: `false`

Data_type: bool, read/write

Specifies whether to keep all existing rows in power domain when moving, resizing or reshaping a power domain on GUI or by command.

Related Commands

- `create_floorplan`

floorplan_max_io_height

`floorplan_max_io_height {true | false}`

Default: `false`

Data_type: bool, read/write

Specifies that the maximum I/O height should be used to calculate a die box with IO placement. By default minimum IO height is used.

Related Commands

- `check_floorplan`

floorplan_minimum_sites

`floorplan_minimum_sites` *`value`*

Default: 0

Data_type: int, read/write

Specifies the number of minimal sites (N) that the `check_floorplan` command will check for the same length site rule. When the `sameLengthSite` option of the `floorplan_check_types` attribute is specified, this attribute is required.

Related Commands

- `check_floorplan`

floorplan_move_macros_with_constraint

`floorplan_move_macros_with_constraint {guide|region|fence|none|all}`

Default: `fence`

Data_type: `enum* {guide region fence none all}`, read/write

Specifies that the macros move with the constraint modules at the same time.

Related Commands

- `check_floorplan`

floorplan_move_preplaced_std_cell_only

`floorplan_move_preplaced_std_cell_only {true|false}`

Default: `fence`

Data_type: `bool`, read/write

Specifies that only the pre-placed standard cells can move with the constraint modules at the same time.

Related Commands

- check_floorplan

floorplan_move_std_cell_with_constraint

`floorplan_move_std_cell_with_constraint {guide|region|fence|none|all}`

Default: `none`

Data_type: `enum* {guide region fence none all}`, read/write

Specifies that standard cells move with the constraint modules at the same time.

Related Commands

- check_floorplan

floorplan_narrow_channel_threshold

`floorplan_narrow_channel_threshold` *`value`*

Default: 0

Data_type: float, read/write

Reports the narrow channels whose width (in micros) is smaller than the specified value. When the `narrowChannel` option of the `floorplan_check_types` attribute is specified, this attribute is required.

Related Commands

- check_floorplan

floorplan_no_cut_row

`floorplan_no_cut_row {true | false}`

Default: `false`

Data_type: bool, read/write

Specifies that the rows are not required to be cut before other operations. The following commands honor this mode setting:

- create_route_blockage_along_track
- check_ufc
- fix_floorplan
- finish_floorplan

With the `floorplan_no_cut_row` attribute set to `false` or not specified, the tool will run above the mentioned commands based on the current row shape. So the rows for objects like insts, place blockages are expected to be cut out before running these commands to get expected results.

If rows are not cut before run these commands are run and the `floorplan_no_cut_row` attribute is set to `true`, the tool will identify the rows that have not been cut before running these commands. It calls the split_row command to get a cut row shape during these commands action. The `split_row` command cuts the rows for insts, hinsts, place blockages, route blockages, power domains, groups, or bumps by default. These commands will work based on the cut row shape. But the actual row shape is not changed after these operations.

Related Commands

- check_floorplan

floorplan_finfet_inst_grid

`floorplan_finfet_inst_grid {`*`lx ly ux uy`*`}`

Default: 0.0 0.0 0.0 0.0

Data_type: rect, read/write

Specifies the FinFET instance grid. The format of the grid is {*x_origin y_origin x_pitch y_pitch*}.

Where,
The *x_origin y_origin* is the lower left corner of the core box. (For example, the core rows).
The *x_pitch* is the width of the default technical site and separates the vertical grid lines.
The *y_pitch* is the FinFET pitch as defined in the LEF and separates the horizontal grid lines.

Note: The values are in microns.

Related Commands

- check_floorpla

floorplan_finfet_manufacturing_grid

`floorplan_finfet_manufacturing_grid {lx ly ux uy}`

Default: 0.0 0.0 0.0 0.0

Data_type: rect, read/write

Specifies the FinFET manufacturing grid. The format of the grid is {*x_origin y_origin x_pitch y_pitch*}.

Where,
The *x_origin y_origin* is the lower left corner of the die box (the design boundary).
The *x_pitch* is the manufacturing grid value and is defined in the LEF.
The *y_pitch* is the FinFET pitch and is defined in the LEF.

Note: The values are in microns.

Related Commands

- check_floorplan

floorplan_finfet_placement_grid

`floorplan_finfet_placement_grid {lx ly ux uy}`

Default: 0.0 0.0 0.0 0.0

Data_type: rect, read/write

Specifies the FinFET placement grid. The format of the grid is {*x_origin y_origin x_pitch y_pitch*}.

Where,
The *x_origin y_origin* is the lower left corner of the die box (the design boundary).
The *x_pitch* is the pitch of lowest vertical metal layer and separates the vertical grid lines.
The *y_pitch* is the FinFET pitch as defined in the LEF and separates the horizontal grid lines.

Note: The values are in microns.

Related Commands

- check_floorplan

floorplan_inst_grid

`floorplan_inst_grid {lx ly ux uy}`

Default: 0.0 0.0 0.0 0.0

Data_type: rect, read/write

Specifies the instance grid. The format of the grid is {*x_origin y_origin x_pitch y_pitch*}.

Where,
The *x_origin y_origin* is the lower left corner of the core box. (For example, the core rows).

The *x_pitch* is the width of the default technical site and separates the vertical grid lines.
The *y_pitch* is the height of the default technical site and separates the horizontal grid lines.

Note: The values are in microns.

Related Commands

- check_floorplan

floorplan_layer_track_grid

floorplan_layer_track_grid {*lx ly ux uy*}

Default: 0.0 0.0 0.0 0.0

Data_type: rect, read/write

Specifies the layer_track grid. The format of the grid is {*x_origin y_origin x_pitch y_pitch*}.

Where,
The *x_origin y_origin* is the lower left corner of the die box (the design boundary).
The *x_pitch* is the pitch of the lowest vertical metal layer and separates the vertical grid lines.
The *y_pitch* is the pitch of the lowest horizontal metal layer and separates the horizontal grid lines.

Note: The values are in microns.

Related Commands

- check_floorplan

floorplan_manufacturing_grid

floorplan_manufacturing_grid {*lx ly ux uy*}

Default: 0.0 0.0 0.0 0.0

Data_type: rect, read/write

Specifies the manufacturing grid. The format of the grid is {*x_origin y_origin x_pitch y_pitch*} from the LEF global MANUFACTURINGGRID statement.

Where,
The *x_origin y_origin* is the lower left corner of the die box (the design boundary).

Note: Verify the layer .mfg_grid values if the mfg_grid varies per layer.

Related Commands

- check_floorplan

floorplan_placement_grid

floorplan_placement_grid {*lx ly ux uy*}

Default: 0.0 0.0 0.0 0.0

Data_type: rect, read/write

Specifies the placement grid. The format of the grid is {*x_origin y_origin x_pitch y_pitch*}.

Where,
The *x_origin y_origin* is the lower left corner of the die box (the design boundary).
The *x_pitch* is the pitch of lowest vertical metal layer and separates the vertical grid lines.
The *y_pitch* is the pitch of lowest horizontal metal layer and separates the horizontal grid lines.

Note: The values are in microns.

Related Commands

- check_floorplan

floorplan_power_rail_layer

`floorplan_power_rail_layer` *string*

Default: " "

Data_type: string, read/write

Specifies the layers used for calculating the power/ground on the bottom attribute of the cell or techsite. By default, the lowest metal layer is used.

Related Commands

- check_floorplan

floorplan_row_site_height

`floorplan_row_site_height {any | odd | even}`

Default: `any`

Data_type: enums {any | odd |even}, read/write

Specifies the odd even row height constraints.

Related Commands

- check_floorplan

floorplan_row_site_width

`floorplan_row_site_width {any | odd | even}`

Default: `any`

Data_type: enums {any odd even}, read/write

Specifies the odd even row width constraints.

Related Commands

- check_floorplan

floorplan_snap_all_corners_to_grid

`floorplan_snap_all_corners_to_grid {true|false}`

Default: `false`

Data_type: bool, read/write

Snaps all corners of an object to the grid. It defines the snapping behavior for the kinds of objects: constraint, ptnCut and black box.

Note: This option only works when the snapping grid is user defined grid or FinFET grid.

- When the snapping grid is ManufactureGrid, the object is snapped to the closest grid.
- When the snapping grid is InstanceGrid and PlacementGrid, the object is moved to first snap the LL corner to the closest location and then to snap the UR corner to closest location.
- When the snapping grid is user defined grid, the object is moved to snap the LL corner to the closest location. Then if the `floorplan_snap_all_corners_to_grid` attribute set to true, the object is moved to snap the UR corner to the closest location.
- When the snapping grid is FinFET grid and the `floorplan_snap_all_corners_to_grid` attribute set to true, all corners of the object are snapped to the closest grid. However, if the `floorplan_snap_all_corners_to_grid` attribute set to false, only the LL corner of the object is snapped to the closest location.

Related Commands

- check_floorplan

floorplan_snap_block_grid

floorplan_snap_block_grid {manufacturing | inst | placement | user_define | layer_track | finfet_inst |finfet_manufacturing | finfet_placement}

Default: manufacturing

Data_type: enum {manufacturing | inst | placement | userdefine | layertrack | finfetInst |finfetmanufacturing | finfetplacement}, read/write

Specifies the block snap rules. It can be set to snap to:

- manufacturing: Manufacture Grid
- inst: Instance Grid
- placement: Placement Grid
- user_define: User-define Grid
- layer_track: layer_track Grid
- finfet_manufacturing: FinFET Manufacture Grid
- finfet_inst: FinFET Instance Grid
- finfet_placement: FinFET Placement Grid

Note: For FinFET designs, the default is finfet_manufacturing

Related Commands

- check_floorplan

floorplan_snap_constraint_grid

floorplan_snap_constraint_grid {manufacturing | inst | placement | user_define | layer_track | finfet_inst |finfet_manufacturing | finfet_placement}

Default: inst, read/write

Data_type: enum {manufacturing inst placement user_define layer_track finfet_inst finfet_manufacturing finfet_placement}, read/write

Specify the constraint snap rules. It can be set to snap to:

- manufacturing: Manufacture Grid
- inst: Instance Grid
- placement: Placement Grid
- user_define: User-define Grid
- layer_track: layer_track Grid
- finfet_manufacturing: FinFET Manufacture Grid
- finfet_inst: FinFET Instance Grid
- finfet_placement: FinFET Placement Grid

Note: For FinFET designs, the default is finfet_inst

Related Commands

- check_floorplan

floorplan_snap_core_grid

`floorplan_snap_core_grid {manufacturing | inst | placement | user_define | layer_track | finfet_inst |finfet_manufacturing | finfet_placement}`

Default: `placement`

Data_type: enum {manufacturing inst placement user_define layer_track finfet_inst finfet_manufacturing finfet_placement}, read/write

Specifies the core snap rules. It can be set to snap to:

- `manufacturing`: Manufacture Grid
- `inst`: Instance Grid
- `placement`: Placement Grid
- `user_define`: User-define Grid
- `layer_track`: layer_track Grid
- `finfet_manufacturing`: FinFET Manufacture Grid
- `finfet_inst`: FinFET Instance Grid
- `finfet_placement`: FinFET Placement Grid

Note: For FinFET designs, the default is `finfet_placement`

Related Commands

- `check_floorplan`

floorplan_snap_die_grid

`floorplan_snap_die_grid {manufacturing | inst | placement | user_define | layer_track | finfet_inst |finfet_manufacturing | finfet_placement}`

Default: `placement`

Data_type: enum {manufacturing inst placement user_define layer_track finfet_inst finfet_manufacturing finfet_placement}, read/write

Specifies the die snap rules. It can be set to snap to:

- `manufacturing`: Manufacture Grid
- `inst`: Instance Grid
- `placement`: Placement Grid
- `user_define`: User-define Grid
- `layer_track`: layer_track Grid
- `finfet_manufacturing`: FinFET Manufacture Grid
- `finfet_inst`: FinFET Instance Grid
- `finfet_placement`: FinFET Placement Grid

Note: For FinFET designs, the default is `finfet_placement`

Related Commands

- `check_floorplan`

floorplan_snap_io_grid

`floorplan_snap_io_grid {manufacturing | inst | placement | user_define | layer_track | finfet_inst |finfet_manufacturing | finfet_placement}`

Default: `manufacturing`

Data_type: enum {manufacturing inst placement user_define layer_track finfet_inst finfet_manufacturing finfet_placement}, read/write

Specifies the IO snap rules. It can be set to snap to:

- `manufacturing`: Manufacture Grid
- `inst`: Instance Grid
- `placement`: Placement Grid
- `user_define`: User-define Grid
- `layer_track`: layer_track Grid
- `finfet_manufacturing`: FinFET Manufacture Grid
- `finfet_inst`: FinFET Instance Grid
- `finfet_placement`: FinFET Placement Grid

Related Commands

- `check_floorplan`

floorplan_snap_place_blockage_grid

`floorplan_snap_place_blockage_grid {manufacturing | inst | finfet_inst | finfet_manufacturing | user_define}`

Default: `inst`

Data_type: enum {manufacturing inst finfet_inst finfet_manufacturing user_define}, read/write

Specifies the placement blockage snap rules.

- `manufacturing`: Manufacture Grid
- `inst`: Instance Grid
- `user_define`: User-define Grid
- `finfet_manufacturing`: FinFET Manufacture Grid
- `finfet_inst`: FinFET Instance Grid

Related Commands

- `check_floorplan`

floorplan_snap_place_blockage_type

`floorplan_snap_place_blockage_type {closest | upsize | sequential}`

Default: `closest`

Data_type: enum {closest upsize sequential}, read/write

Specifies a snapping type for placement blockages. The snapping types can be:

- `closest`: The tool snaps all corners of the placement blockage to the closest specified grid, regardless of which direction could increase or decrease the blockage size.
- `upsize`: The tool snaps all corners of the placement blockage to the corresponding closest specified grid in the direction which would increase the objects' size.
- `sequential`: The tool moves the placement blockage to snap the low-left corner to the closest specified grid, then snaps the upper-right corner to the closest grid.

This setting is honored by the following commands:

- `update_floorplan_obj`
- `create_place_blockage {-rects|-polygon}`
- `snap_floorplan -place_blockage`

Related Commands

- check_floorplan

floorplan_user_define_grid

`floorplan_user_define_grid {x_offset y_offset x_pitch y_pitch}`

Default: 0.0 0.0 0.0 0.0

Data_type: rect, read/write

Specifies the user-defined grid. The format is {*x_offset y_offset x_pitch y_pitch*}.

Where,
x_offset y_offset is the offset of the user-defined grid.
x_pitch separates the vertical grid lines.
y_pitch separates the horizontal grid lines.

Note: The values are in microns.

Related Commands

- check_floorplan

floorplan_vertical_row

`floorplan_vertical_row {0 | 1}`

Default: 0

Data_type: int, read/write

Specifies whether the rows in the floorplan are vertical (1) or horizontal (0). By default, the rows in the floorplan are horizontal. Set this variable to `1` to enable vertical rows.

Related Commands

- init_design

get_draw_view

```
get_draw_view
[-help]
```

Returns the design view that was set with the `gui_set_draw_view` command. You can use this command after `gui_set_draw_view`.

Parameters

`-help`	Outputs a brief description of the `get_draw_view` command. For a detailed description of the command, use the `man` command: `man get_draw_view`

Related Information

- `gui_set_draw_view`
- Floorplanning the Design chapter in the *Innovus Stylus Common UI User Guide*

get_io_flow_flag

```
get_io_flow_flag
[-help]
```

Gets the current I/O row flow flag setting. To set the I/O row flow flag, use the `set_io_flow_flag` command.

Parameters

`-help`	Outputs a brief description of the `get_io_flow_flag` command. For a detailed description of the command, use the man command: `man get_io_flow_flag`

Example

- The following command displays the current I/O row flow flag settings:
  ```
  get_io_flow_flag
  ```

Related Information

- `create_io_row`
- `set_io_flow_flag`
- Floorplanning the Design chapter in the *Innovus Stylus Common UI User Guide*

get_macro_place_constraint

```
get_macro_place_constraint
[-help]
[-macro_corner_keepout]
[[-array array_name] |
[-inst string  -orientation ] |
[-cell string  -cell_obs ] |
[-track_adjustment ] |
[-max_io_pin_group_keep_out ] |
[-forbidden_space_to_core ] |
[-forbidden_space_to_macro ] |
[-min_space_to_core ] |
[-min_space_to_macro ] |
[-parallel_run_length ] |
[-align_group string ] |
[-horizontal_stacking ] |
[-vertical_stacking ] |
[-honor_strict_spacing_constraint ] |
[-avoid_abut_macro_edge_with_pins ] |
[-same_length_site ] |
[-pg_resource_model ] |
[-non_default ] |
[-halo_sharing ]]
```

Gets the current value of the macro constraints that are honored by the macro placer. To define the constraints of macros, use the `set_macro_place_constraint` command.

Note: The report displayed when the -array, -cell/-cell_obs, or -inst/-orientation parameters are used can result in long lines on the screen that are supported by ellipses (…) indicating that more information about the macro constraint follows. If you want to get all the information together, Cadence recommends using the `write_macro_place_constraint` command to save all the information into a file.

Parameters

`-align_group string`	Specifies the align group name. *Data_type*: string, optional
`-array array_name`	Specifies the macro array name. *Data_type*: string, optional
`-avoid_abut_macro_edge_with_pins`	When specified, it returns if abutting macro edges with pins must be avoided even if the macro distance is within the forbidden space setting *Data_type*: bool, optional
`-cell_obs string`	When specified, it returns the cell obstruction value. *Data_type*: bool, optional
`-cell string`	Specifies constraints for all instances of the specified cells. *Data_type*: string, optional
`-forbidden_space_to_core`	When specified, it returns the forbidden space to core value. *Data_type*: bool, optional
`-forbidden_space_to_macro`	When specified, it returns the forbidden space to macro value. *Data_type*: bool, optional
`-halo_sharing`	When specified, it returns the halo sharing mode. *Data_type*: bool, optional
`-help`	Outputs a brief description that includes the type and default information for each `get_macro_place_constraint` parameter. For a detailed description of the command and all of its parameters, use the man command: `man get_macro_place_constraint`

`-honor_strict_spacing_constraint`	When specified, it returns if the forbidden spacing and minimum spacing constraints must be honored strictly. *Data_type*: bool, optional
`-horizontal_stacking`	When specified, it returns the horizontal stacking constraint. *Data_type*: bool, optional
`-inst` *string*	Specifies the macros on which the constraints are set. *Data_type*: string, optional
`-macro_corner_keepout`	When specified, it returns the macro corner keepout value *Data_type*: bool, optional
`-max_io_pin_group_keep_out`	When specified, it returns the max height of the I/O pin blockages. *Data_type*: bool, optional
`-min_space_to_core`	When specified, it returns the minimum space to core value. *Data_type*: bool, optional
`-min_space_to_macro`	When specified, it returns the minimum space to macro value. *Data_type*: bool, optional
`-non_default`	When specified, it displays the current settings which are not the default settings. *Data_type*: bool, optional
`-orientation`	When specified, it returns the valid orientations. *Data_type*: bool, optional
`-parallel_run_length`	When specified, it returns the parallel run length value. *Data_type*: bool, optional
`-pg_resource_model`	When specified, it returns the pg resource model details. *Data_type*: bool, optional
`-same_length_site`	When specified, it returns the the minimum number of sites between the vertical edge of two macros. *Data_type*: bool, optional
`-track_adjustment`	When specified, it returns the track adjustment value. *Data_type*: bool, optional
`-vertical_stacking`	When specified, it returns the vertical stacking constraint. *Data_type*: bool, optional

Example

- The following command gets the current value of the macro constraints that are honored by the macro placer.:
  `get_macro_place_constraint`

Related Information

- `check_macro_place_constraint`
- `create_pg_model_for_macro_place`
- `reset_macro_place_constraint`
- `set_macro_place_constraint`
- `write_macro_place_constraint`
- Floorplanning the Design chapter in the *Innovus Stylus Common UI User Guide*
- Using the Mixed Placer chapter in the *Innovus Stylus Common UI User Guide*

gui_set_draw_view

```
gui_set_draw_view
[-help]
mode
```

Sets the design view in the design display area. You can use this command after importing or restoring the design. To see the current view use `get_draw_view`.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `gui_set_draw_view` parameter. For a detailed description of the command and all of its parameters, use the man command: `man gui_set_draw_view`
mode	Sets the mode. • `amoeba`: Sets the view to Amoeba view. This view displays the outline of the modules and submodules after placement, showing physical locality of the module. • `fplan`: Sets the view to floorplan view. This view displays the hierarchical module and block guides, connection flightlines, and floorplan objects, including block placement, and power and ground nets. • `place`: Sets the view to physical view. This view displays the detailed placements of the module's blocks, standard cells, nets, and interconnects. *Data_type*: string, required

Related Information

- `get_draw_view`
- Floorplanning the Design chapter in the *Innovus Stylus Common UI User Guide*

gui_show_hinst_in_floorplan

```
gui_show_hinst_in_floorplan
[-help]
name
[-add_or_remove {0|1}]
```

Use this command to display modules in the floorplan view whose instance count is smaller than the `set_preference MinFPModuleSize` value. By default, modules whose instance count is less than `MinFPModuleSize` are not displayed (default is 1). The `MinFPModuleSize` variable can be set to a larger value to focus only on larger modules which require floorplanning. You can then display specific smaller modules using this command. For example, use `gui_show_hinst_in_floorplan` if you need to interactively floorplan small regions or fences to control placement of small modules of logic.

Note: You can change the *Min. floorplan Module Size* option in the *Preferences* form in the *Display* menu and import the design. Alternatively, you can use the `set_preference MinFPModuleSize value` command.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `gui_show_hinst_in_floorplan` parameter. For a detailed description of the command and all of its parameters, use the man command: `man gui_show_hinst_in_floorplan`
name	Specifies the name of the hierarchical instance. *Data_type*: string, required
`-add_or_remove {0|1}`	Adds or removes the specified hierarchical instance. Specify 1 to add or 0 to remove. *Data_type*: int, optional

Example

- The following command adds hierarchical instance SH99 in the floorplan view:
  ```
  gui_show_hinst_in_floorplan SH99 1
  ```
- The following command only displays modules with 200 instances or more:
  ```
  set_preference MinFPModuleSize 200
  ```
  If module mcore0/a0 has less than 200 instances you can display it using:
  ```
  gui_show_hinst_in_floorplan mcore0/a0 1
  ```
  To disable its display:
  ```
  gui_show_hinst_in_floorplan mcore0/a0 0
  ```

Related Information

- `set_preference`
- Floorplanning the Design chapter in the *Innovus Stylus Common UI User Guide*

gui_show_obj_connectivity

```
gui_show_obj_connectivity
[-help]
[-direction {in|out|all}]
[-level level]
[-line_to_ports]
[-min_connection number]
{-insts string | -ports string | -selected | -reset }
[-through_registers [-all_registers ]]
```

Displays the connections of specified macros, selected macros, or specified ports. You can also display the input and output connectivity separately. By default it displays both.

Parameters

`-all_registers`	Displays all registers found during the analysis. *Data_type*: bool, optional
`-help`	Outputs a brief description that includes type and default information for each `gui_show_obj_connectivity` parameter. For a detailed description of the command and all of its parameters, use the man command: `man gui_show_obj_connectivity`
`-direction {in|out|all}`	Specifies the pin direction of objects to be displayed. *Default*: `all` *Data_type*: string, optional
`-insts string`	Specifies the macros for which the connectivity is displayed. *Data_type*: string, optional
`-level level`	Specifies the trace level to display. *Maximum Level*: 10 *Minimum Level*: 1 *Default*: 1 *Data_type*: int, optional
`-line_to_ports`	Displays the display paths from/to primary ports. *Data_type*: bool, optional
`-min_connection number`	Specify the trace minimum connection. *Default*: 1 *Data_type*: int, optional
`-ports string`	Specifies the IO ports for which the connectivity is displayed. *Data_type*: string, optional
`-reset`	Clears the trace macro display. *Data_type*: bool, optional
`-selected`	Displays the connectivity of selected macros. *Data_type*: bool, optional
`-through_registers`	Displays the flight lines going through registers. *Data_type*: bool, optional

Example

The following command displays the connections for blk1 upto level 3:

```
gui_show_obj_connectivity -insts blk1 -level 3
```

Related Information

- `trace_obj` Category Attributes
- `trace_obj_connectivity`
- Floorplan Menu chapter in the *Innovus Stylus Common UI Menu Reference*

init_core_rows

```
init_core_rows
[-help]
[-power_domain string]
```

Regenerates rows for the core area and the power domains in a design with rows based on the current row parameters instead of using the `create_floorplan` command. You can use this command after importing the design.

Note: In a common flow, Innovus goes over all technical sites used in the netlist and creates rows for these sites. It chooses the shortest tech site in the design (with the smallest height and width) as the default-technical-site. All other used tech-sites with integer-multiple height/width are considered as compatible.

Parameters

`-help`	Outputs a brief description of the `init_core_rows` command. For a detailed description of the command, use the `man` command: `man init_core_rows`
`-power_domain string`	Specifies the power domain for which rows are to be created. *Data_type*: string, optional

Related Information

- `create_floorplan`
- `create_row`
- `delete_row`
- `split_row`
- `stretch_rows`
- Floorplanning the Design chapter in the *Innovus Stylus Common UI User Guide*

legalize_floorplan

```
legalize_floorplan
[-help]
[-check_orient]
[-check_site]
```

Legalizes partition locations according to standard cell row orientations and specific design constraints. You can use this command after specifying the floorplan.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `legalize_floorplan` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man legalize_floorplan`
`-check_orient`	Specifies that all partitions snap to the closest `R0` row on the grid. For this rule, the Innovus software assumes that standard cell rows start with an `R0` orientation and continue in an `R0`-to-`MX` pattern. *Data_type*: bool, optional
`-check_site`	Specifies that partitions be placed on grids so that they are legal for all design constraints. *Data_type*: bool, optional

Example

The following command legalizes all the partition locations such that they are snapped to the closest `R0` rows:

```
legalize_floorplan -check_orient
```

Related Information

- `check_floorplan`
- Floorplanning the Design chapter in the *Innovus Stylus Common UI User Guide*

move_group_pins

```
move_group_pins
[-help]
[-depth float]
[-end_location {x y}]
[-expand_spacing_factor float]
[-layer string] -location {x y}
[-no_fixed]
[-spacing float]
[-spread_type string]
[-start_location {x y}]
[-width float]
[-with_overlap]
```

Changes the pin layer, the pin size, pin status, resolves pin overlap, and moves a selected pin or pin group to a specific location. You can use this command after selecting the pin(s) in the move mode (selecting pins in the design display area with the *Move/Resize/Reshape* tool widget).

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `move_group_pins` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man move_group_pins`
`-depth float`	Specifies the depth of the pin or pin group, in micrometers. *Data_type*: float, optional
`-end_location {x y}`	Specifies the coordinates where the pins should end when the pins are spread along a range. *Data_type*: point, optional
`-expand_spacing_factor float`	Specifies the value that will be multiplied to the current spacing in tracks. *Data_type*: float, optional
`-layer string`	Specifies the number of the metal layer for the pin or pin group. *Data_type*: string, optional
`-location {x y}`	Specifies the new x and y location for the pin or pin group. *Data_type*: point, required
`-no_fixed`	Specifies that the moved pins or group of pins are not set to the *fixed* status. *Data_type*: bool, optional
`-spacing float`	Specifies the spacing between pins for multi-pin editing. *Data_type*: float, optional
`-spread_type string`	Specifies how spacing between pins is managed while moving grouped-pins. *Data_type*: string, optional
`-start_location {x y}`	Specifies the coordinates where the pins should start when the pins are spread along a range. *Data_type*: point, optional
`-width float`	Specifies the width of the pin or pin group, in micrometers. *Data_type*: float, optional
`-with_overlap`	Specifies that moved pin(s) can overlap. *Data_type*: bool, optional

Example

The following command moves the selected pin group to a new location (10,20):

```
move_group_pins -location 10 20
```

Related Information

- Floorplanning the Design chapter in the *Innovus Stylus Common UI User Guide*
  - Editing Pins

move_macro_in_hinsts

```
move_macro_in_hinsts
[-help]
{-hinsts {hinst_name(s)} | -all}
```

Moves all hard macros that belong to the specified module into its boundary. You use this command after the module is placed in the core area.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `move_macro_in_hinsts` parameter. For a detailed description of the command and all of its parameters, use the man command: `man move_macro_in_hinsts`.
`-all`	Moves all hard macros that belong to hierarchical modules. *Data_type*: bool, optional
`-hinsts {hinst_name(s)}`	Specifies the name of the hierarchical instance. You can use wildcards (*?) with this parameter. *Data_type*: string, optional

Example

The following command moves all hard macros that belong to modules that have hierarchical instance names that match DTMF_INST/*_TEST_INST into its boundary:

```
move_macro_in_hinst -hinsts DTMF_INST/*_TEST_INST
```

Related Information

- Floorplanning the Design chapter in the *Innovus Stylus Common UI User Guide*

move_obj

```
  move_obj
[-help]
[-group]
{ -point {x y} | { -direction {up down left right} { -distance value | -to {core_box die_box} }}}
{ obj_list | -selected }
```

Moves objects (instances, modules, groups, placement blockage, routing blockage, hard macros, special routings, and I/O pins) vertically or horizontally by a specified distance. The objects can be moved to the die or core box boundary, as specified. You can use this command after selecting one or more objects in the design display area.

Parameters

`-direction {up down left right}`	
	Specifies the vertical and horizontal movement for the selected objects. Choose one of the following: • `up`: Specifies an upward move. • `down`: Specifies a downward move. • `left`: Specifies a left move • `right`: Specifies a right move. *Data_type*: enum, optional
`-distance` *value*	Specifies the distance, in micrometers, that the objects will be moved in the specified direction. *Data_type*: float, optional
`-group`	Moves all the specified or selected objects as a group. *Data_type*: bool, optional
`-help`	Outputs a brief description that includes the type and default information for each `move_obj` parameter. For a detailed description of the command and all of its parameters, use the man command: `man move_obj`
obj_list	Specifies the list of objects to move. *Data_type*: (any_object)+, optional
`-point {`*x y*`}`	Specifies the lower left point that the objects will be moved to. *Data_type*: point, optional
`-selected`	Moves the selected object. *Data_type*: bool, optional
`-to {core_box die_box}`	
	Moves the object to the core_box or the die_box boundary, as specified. *Data_type*: enum, optional

Examples

- The following command moves each of selected objects to the left of core boundary:
  ```
  move_obj -selected -direction left -to core_box
  ```
- The following command moves the bounding box of all selected objects to the left of core boundary:
  ```
  move_obj -selected -direction left -to core_box -group
  ```
- The following command moves each of the selected objects to the left with 50um. The same as with option -group:
  ```
  move_obj -selected -direction left -distance 50
  ```
- The following command moves each of the selected objects' low left coordinates to the specified point:
  ```
  move_obj -selected -point {330 410}
  ```
- The following command moves the bounding box's low left coordinates of all selected objects to the specified point:
  ```
  move_obj -selected -point {330 410} -group
  ```

Related Information

- `copy_obj`
- `paste_obj`
- `unplace_obj`
- Floorplanning the Design chapter in the *Innovus Stylus Common UI User Guide*

paste_obj

```
paste_obj
[-help]
```

Pastes copied placement or routing blockages. Use the `copy_obj` command to copy the required object. Use this command any time after floorplanning the design. You can also copy and paste selected placement and routing blockages from the GUI by using the Copy tool. To access the Copy tool, choose *Edit - Copy* from the menu bar or press `F3` after clicking the *Copy* widget ( ) on the toolbar.

Parameters

`-help`	Prints a brief description of the `paste_obj` command. For a detailed description of the command, use the `man` command: `man paste_obj`

Example

- The following commands copy and paste the selected placement and routing blockages:
  ```
  copy_obj
  paste_obj
  ```

Related Information

- `copy_obj`
- `edit_copy`
- `move_obj`
- Floorplanning the Design chapter in the *Innovus Stylus Common UI User Guide*

place_area_ios

```
place_area_ios
[-help]
[-only_area_ios]
[-assign_bumps]
[-max_distance distance]
[-fast]
[-packing]
[-ignore_area_ios {list}]
[-ignore_area_io_cells {list}]
[-hard_fence]
```

Places I/O driver cells. The area I/O Library must be loaded when importing the design. Area I/O rows are also required.

Note: Once you run the `place_area_ios -only_area_ios` command to place the area I/Os, you can run `place_peripheral_ios -cells` to specify the cell(s) that must be placed on the periphery.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `place_area_ios` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man place_area_ios`
`-assign_bumps`	If you have unassigned bumps for area I/O instance connections, this connects area I/O instances to the nearest unassigned bumps. This option requires that bump cells and bump locations are already loaded. **Note:** You can also assign bumps after area I/O placement by using the `assignBump` command. *Data_type*: bool, optional
`-fast`	Runs area I/O placement in the fast mode, trading speed for quality. *Data_type*: bool, optional
`-hard_fence`	If specified, restricts the area I/O placement to the power domain fence to which the area I/O's belong to. *Data_type*: bool, optional
`-ignore_area_io_cells` {*list*}	Ignores all area I/O instances that instantiate from the cells specified in the list, during placement. For example, if a cell `PDI` has 2 AIO instances, `test_in_1` and `test_in_2`, this parameter ignores both the instances during placement. *Data_type*: string, optional
`-ignore_area_ios` {*list*}	Ignores all the area I/O cells specified in the list, during placement. Specify this parameter if you already placed the area I/O cells on the periphery and you want to avoid placing the same cells in the core. *Data_type*: string, optional
`-max_distance` *value*	Specifies the maximum allowable distance between the I/O driver cells and the bumps. *Data_type*: int, optional
`-only_area_ios`	Places only area I/O instances. If this option is not specified, the command will also place all standard cell instances and blocks. *Data_type*: bool, optional
`-packing`	Specifies the packing of I/O driver cells within each cluster of rows. This option is only applicable to designs with clusters of I/O rows. *Data_type*: bool, optional

Examples

- The following command places area I/O instances and all standard cell instances, and assign bumps for all chip I/O pins:
  ```
  place_area_ios -assign_bumps
  ```
- The following command places only area I/O cells:
  ```
  place_area_ios -only_area_ios
  ```

Related Information

- `place_peripheral_ios`
- Floorplanning the Design chapter in the *Innovus Stylus Common UI User Guide*

place_blocks_refine

```
place_blocks_refine
[-help]
[[-area {x1 y1 x2 y2} [-selected]]
{-area {x1 y1 x2 y2} | -permute_pack | -mark_step | -read_step step_number
       |-read_mark mark_number}
[-pack_adjustment]
[-io_pin_clearance]]
```

Refines the placement of macros in the floorplan. By default, the `place_blocks_refine` command performs global incremental macro adjustment. You can also adjust specific macro packs, or all of the macros in a specific area of the design. You can use `place_blocks_refine` command after creating an initial floorplan and analyzing the results. Macros must have a status of PLACED in the database in order to use this command.

Parameters

`-pack_adjustment`	Adjusts the way that macros packs are placed, relative to their locations. For example, an array of 16 macros might be placed in four rows of four macros (4 x 4), if they are near the top or bottom of the chip. This might change to eight rows of two macros (8 x 2), if the macros are moved near the left or right edge of the chip.
`-area` *x1 y1 x2 y2*	Adjusts the placement of any macros within the specified area to reduce empty area between them. If you also specify `-selected`, the `place_blocks_refine` command moves the selected macros to the specified area and adjusts their position along with any macros already within the area. *Data_type*: box, optional
`-help`	Outputs a brief description that includes type and default information for each `place_blocks_refine` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man place_blocks_refine`.
`-io_pin_clearance`	Leaves an internally calculated amount of clearance space between placed macros and block I/O pins. Use this parameter to ensure that pins are accessible to the router. *Data_type*: bool, optional
`-mark_step`	Marks an intermediate macro refinement step to save. When you perform incremental macro refinement, each adjustment is considered a step. By default, all incremental steps done with the `place_blocks_refine` command are saved and listed in the log file with a number. Since the manal refinement steps (such as move, rotate, and flip) are not saved by default, you can use this parameter to mark manual refinement steps, or any other important step that you want to save. **Note:** Innovus saves up to 1000 incremental refinement steps in the database and 20 marks before overwriting them. *Data_type*: bool, optional
`-permute_pack`	Adjusts the placement of a selected pack of macros to reduce empty area between them. A pack is a set of macros from the same seed that have similar sizes and aspect ratios and have been grouped together. You must select the macro pack in the main window before using this parameter. You can select one macro to select the entire pack. Macro pack information is stored temporarily in the database. If you quit the Innovus session, the information is lost. *Data_type*: bool, optional
`-read_mark` *mark_number*	Returns the design to the state it was in after the specified marked step (`-mark_step`). For example, to return the design to the state it was in after the step saved as `mark2`, specify: `place_blocks_refine -read_mark 2` You can find the mark numbers listed in the log file. *Data_type*: int, optional

`-read_step` *step_number*	Returns the design to the state it was in after the specified intermediate non-marked step. For example, to return the design to the state it was in after step 4, specify:
	`place_blocks_refine -read_step 4`
	To return the design to the state it was in after the previous adjustment, specify:
	`place_blocks_refine -read_step -1`
	You can find the step numbers listed in the log file.
	Data_type: int, optional

`-selected`	Moves the selected macros to the specified area and adjusts their position along with any macros already within the area. **Note:** You cannot use this parameter without specifying the `-area` parameter. *Data_type*: bool, optional

Example

The following command moves the selected macros to a new area:

```
place_blocks_refine -selected -area 100 100 200 300
```

Related Information

- Floorplanning the Design chapter in the *Innovus Stylus Common UI User Guide*

place_macro_detail

```
place_macro_detail
[-help]
[-verbose ]
```

Legalizes the macros based on constraints such as halo and blockages, forbidden spacing, min-space, etc. To define the constraints of macros, use the `set_macro_place_constraint` command.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `place_macro_detail` parameter. For a detailed description of the command and all of its parameters, use the man command: `man place_macro_detail`
`-verbose`	Writes the macro movement details to a tcl file. *Data_type*: bool, optional

Example

The following command specifies the minimum space between macro-to-core boundary as 5 and the forbidden space between macro-to-core boundary as 2.

```
set_macro_place_constraint -forbidden_space_to_core 2 -min_space_to_core 5
place_macro_detail
```

Related Information

- `set_macro_place_constraint`
- Floorplanning the Design chapter in the *Innovus Stylus Common UI User Guide*
- Using the Mixed Placer chapter in the *Innovus Stylus Common UI User Guide*

place_pad_ios

```
place_pad_ios
[-help]
[-row_number number_of_rows]
[-max_io_height]
```

Places the I/O pads evenly from one row to multiple rows. You can use this command after design initialization when the Innovus software automatically places the I/O pads in one row.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `place_pad_ios` parameter. For a detailed description of the command and all of its parameters, use the man command: `man place_pad_ios.`
`-row_number number_of_rows`	Specifies the number of I/O rows needed for the placement. *Data_type*: int, optional
`-max_io_height`	Specifies the maximum height used to calculate the head box. By default, this command uses the minimum height. *Data_type*: bool, optional

Example

The following command places I/O pads in two I/O rows:

```
place_pad_ios -row_number 2
```

Related Information

- Floorplanning the Design chapter in the *Innovus Stylus Common UI User Guide*

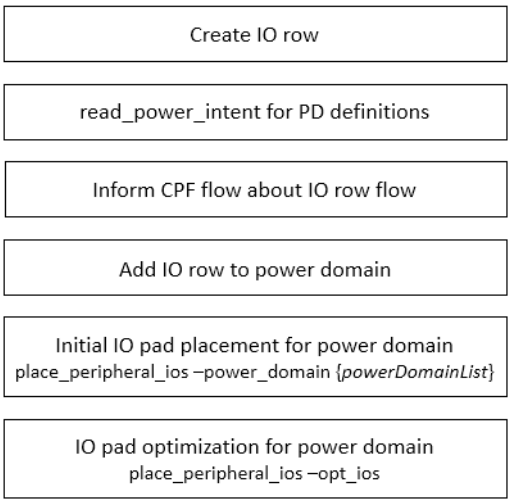
place_peripheral_ios

```
place_peripheral_ios
[-help]
[-assign_bump]
[-cells {cellNamePattern}]
[-extra_config fileName]
[-insts {instNamePattern}]
[-io_file fileName]
[-max_io_height]
[-no_random_placement]
[-opt_ios]
[-overflow_map]
[-power_domain powerDomainName]
[-route_constraint_file fileName]
```

Places CLASS PAD AREAIO cells on the die boundary (periphery) in random order. Use this command after the design is loaded.

- The command, by default, honors pre-placed or `FIXED` area I/Os in the design and does not modify their placement when executed.
- The command can place IO pads onto the IO rows that belong to a power domain and optimize the locations of the IO pads for the power domain.
  ```
  place_peripheral_ios -power_domain {VCCQ_MC0 VCCQ_U VCCQ2_E VCCQ_L VCCQ2_H VCCQ_D VCCQ2_C}
  place_peripheral_ios -opt_ios
  ```

  The I/O row information is retrieved from the power domain itself. IO rows need to be created and added to power domain before the placement and optimization. Run `place_peripheral_ios -power_domain {powerDomainName}` for a random placement then use `place_peripheral_ios -opt_ios` for optimized result.

  You can use this function to place IO pads in a design that has multiple power domains with IO pads. Innovus recognizes the relationship between the IO pads, IO rows, and power domains and so placement and optimization are automatic.
- The command reads options specified by the `flip_chip` category attributes before placing the CLASS PAD AREA IO cells on the die boundary. It retrieves the constraint file and extra configuration file information from the `flip_chip_constraint_file` and `flip_chip_extra_config` attributes.
- During bump assignment, the command uses resistance constraints that are specified in the constraint file for net(s).
- The `place_peripheral_ios -opt_ios -assign_bump` command also reads the `flip_chip_route_width` value before reassigning the signal bumps in the design.
- When the I/O row flow is enabled (`set_io_flow_flag 1`), the `place_peripheral_ios` command restricts I/O placement in the rows created previously, instead of all possible locations. However, you need to ensure that the I/O rows are created outside the I/O box.
  Command flow:
  ```
  set_io_flow_flag 1
  create_io_row
  ```

```
place_peripheral_ios
```

Note: IO cells must be defined as CLASS PAD on the boundary of the design. The peripheral I/O Library must be loaded when importing the design. Peripheral I/O rows are also required.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `place_peripheral_ios` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man place_peripheral_ios`.
`-assign_bump`	Allows signal bumps to be reassigned to improve routing if the I/O cells have been fixed. *Data_type*: bool, optional
`-cells {cellNamePattern}`	Specifies which area I/O cell(s) must be placed on the periphery. Once you run `place_peripheral_ios -cells` to grab the area I/O cells for placement on the periphery, you can run `place_area_ios -ignore_area_ios {list}` to avoid the placement of these cells in the core area. *Data_type*: string, optional
`-extra_config fileName`	Specifies a file containing user-defined placement options written in `route_special` syntax. *Data_type*: string, optional
`-insts {instNamePattern}`	Specifies which area I/O instance(s) must be grabbed for placement on the periphery. Once you run `place_peripheral_ios -insts` to place the area I/O instances on the periphery, you can run `place_area_ios -ignore_area_ios {list}` to avoid the placement of these instances in the core area. *Data_type*: string, optional
`-io_file fileName`	Specifies the name of a file to control the placement of the I/O cells. Supports the version 3.0 I/O file format. *Data_type*: string, optional
`-max_io_height`	Specifies the maximum height for the I/O row. *Data_type*: bool, optional
`-no_random_placement`	Without options, the `place_peripheral_ios` command randomly places all I/O cells on the periphery. This option inhibits this behavior and optimizes pads without initial placement. *Data_type*: bool, optional
`-opt_ios`	Allows I/O cells to be moved to improve routing if the bumps have been fixed, optimizing the initial placement. *Data_type*: bool, optional

`-overflow_map`	Displays the congestion map using the width and spacing parameters specified in the extra configuration file. To display the congestion map, run the following command to specify the options for the extra configuration file:
	`place_peripheral_ios -no_random_placement -overflow_map -extra_config scripts/global.cfg`
	where:
	`scripts/global.cfg:`
	`srouteBottomLayerLimit 8 /* Assume the intended routing layer range is M8 to M9 */`
	`srouteTopLayerLimit 9`
	`srouteRouteWidth 10000 /* n is the routing width, in db unit - 5u */`
	Data_type: bool, optional

`-power_domain powerDomainName`	Specifies the name of the power domain that consists of peripheral I/Os. The I/O row information can now be retrieved from the power domain itself. *Data_type*: string, optional
`-route_constraint_file fileName`	Specifies a file with constraints to control the placement of the I/O cells. The command uses resistance constraints specified in this file, for cell placement. *Data_type*: string, optional

Example

- The following command places CLASS PAD AREA IO cells on the die boundary (periphery) and allows I/O cells to be moved to improve routing if the bumps have been fixed, optimizing the initial placement:

```
place_peripheral_ios -opt_ios
```

Related Information

- `place_area_ios`
- Tools Menu chapter in the *Innovus Stylus Common UI Menu Reference*
  - Flip Chip Route - Advanced - Routing Constraints GUI
- Flip Chip Methodologies chapter in the *Innovus Stylus Common UI User Guide*
  - Useful Constraints for Flip Chip Routing

plan_design Category Attributes

The root attributes in the `plan_design` category are:

- plan_design_domain
- plan_design_rail_model
- plan_design_total_power

To set the value of any root attribute, use the `set_db` command:

```
set_db attribute_name value
```

To get the current value of any root attribute, use the `get_db` command:

```
get_db attribute_name
```

To get the current value of all attributes in the `plan_design` category, use the following command:

```
get_db -category plan_design
```

plan_design_domain

`plan_design_domain {{pd1 float net1} {pd2 float net2}..}`

Default: `" "` (empty string)

Data_type: string, read/write

Specifies the power, in milliwatts, assigned for nets in power domains.

Note: If the total power or the domain power is not specified, the `plan_design_domain` attribute will estimate it internally based on the instance area.

Related Commands

- `proto_design`

plan_design_rail_model

`plan_design_rail_model {virtual | existing}`

Default: `existing`

Data_type: enum {virtual existing}, read/write

Specifies the power grid to use.

- `virtual`: Use routing blockages to model virtual power grid. These routing blockages are removed after congestion analysis is done.
- `existing`: Use the current power grid.

Related Commands

- `proto_design`

plan_design_total_power

`plan_design_total_power float`

Default: `0`

Data_type: double, read/write

Specifies a target value for the power, in milliwatts, consumed by the design.

Related Commands

- `proto_design`

read_floorplan

```
read_floorplan
file
[-help]
[-no_equalize_partition_hinst]
[-sections {macro | pin | bndry | special_route | pin_constraint}}]
[-xml]
```

Loads a floorplan file. You can use this command after importing the design.

Note:

- When you run the `read_floorplan` command, the Innovus software removes any existing floorplan information and reads in the new information. The `read_floorplan` command honors the blackbox min/max aspect ratio information in the floorplan information file.
- Blocks and instances loaded with this command are set as preplaced.
- The `read_floorplan` command loads the user-specified seed information by default.
- The `loadFPlan` command loads the information related to blockages, tracks, and pin shapes from the *.fp file.

Parameters

file	Specifies the name of the floorplan file that was saved. *Data_type*: string, required
`-help`	Outputs a brief description that includes type and default information for each `read_floorplan` parameter. For a detailed description of the command and all of its parameters, use the man command: `man read_floorplan.`
`-no_equalize_partition_hinst`	Disables the snapping capability of the clone partitions to the power grid, such that clones do not necessarily have the same row structure and pattern as their master. *Data_type*: bool, optional
`-sections {macro | pin | bndry | special_route | pin_constraint}`	
	Specifies the objects section in the floorplan file that will be updated. The sections can be macro, pin, brndy, special_route, pin_constraint. **Note:** The `read_floorplan` command supports routing blockages on master slice layers. *Data_type*: string, optional
`-xml`	Reads the floorplan information from the XML format floorplan file. *Data_type*: bool, optional

Example

- The following command loads the floorplan file myFPlan.fp, which was saved in an earlier session:
  ```
  read_floorplan myFPlan.fp
  ```
- The following command loads only the pin constraints data in the present design with other floorplan objects remaining unchanged.
  ```
  read_floorplan myFPlan.fp -sections {pin_constraint}
  ```

Related Information

- `create_floorplan`
- `write_floorplan`
- `floorplan` Category Attributes
- Floorplanning the Design chapter in the *Innovus Stylus Common UI User Guide*
  - Module Constraint Types

- Viewing the Floorplan

read_io_file

```
read_io_file
[-help]
file_name
[-no_die_size_adjust]
[-specified_ios_only]
[-eco [-pads_only | -bumps_only]]
```

Loads an I/O assignment file. You can use this command after importing the design.

Notes:

- By default, the `read_io_file` command will automatically adjust the die size to accommodate all the IOs. If this command is used after floorplanning, it would change the die size defined in the floorplan file. If you do not want the die size to be automatically adjusted, use the `-no_die_size_adjust` parameter.
- During I/O inst placement, the *space* value determines the distance between the current inst and the previous inst. Earlier, while honoring `iopad` spacing, the `read_io_file` command considers the previous inst to be the inst that appears before the current inst when insts are placed according the io_order specified in the I/O assignment file .

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `read_io_file` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man read_io_file`
`-bumps_only`	Loads only bump ECO operations in specified IO file. *Data_type*: bool, optional
`-eco`	Allows I/O ECO change by loading a new I/O file. In a chip-package co-design situation, when I/O Planner (IOP) utility in SiP is used for modifying the I/O and bump information, the modified information needs to be passed to Innovus. When `-eco` is specified, `read_io_file` overwrites the existing setting for I/Os and bumps with the information from the new I/O file. *Data_type*: bool, optional
file_name	Specifies the name of the I/O assignment file. *Data_type*: string, required
`-no_die_size_adjust`	Specifies that the die size should not be adjusted automatically. *Data_type*: bool, optional
`-pads_only`	Loads only IO pad ECO operations in specified IO file. *Data_type*: bool, optional
`-specified_ios_only`	Loads only the IO in the specified IO file. This parameter enables you to place IO cells listed in the I/O assignment file without deleting or moving the IO cells not listed in I/O assignment file. *Data_type*: bool, optional

Example

The following command loads the I/O assignment file `myPinFile.io`:

```
read_io_file myPinFile.io
```

Related Information

- Floorplanning the Design chapter in the *Innovus Stylus Common UI User Guide*

report_floorplan_obj

```
report_floorplan_obj
[-help]
[-power_domain]
```

Prints the properties of selected objects to the console. This command supports only instances and modules. You can use this command after selecting one ore more instances or modules.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `report_floorplan_obj` parameter. For a detailed description of the command and all of its parameters, use the man command: `man report_floorplan_obj`
`-power_domain`	Use this option to output power domain information, as well as check if the objects belong to a power domain. *Data_type*: bool, optional

Example

The following commands select and print property information for module `modA`:

```
select_obj [get_db insts modA]
report_floorplan_obj -power_domain
```

Related Information

- `select_obj`
- Floorplanning the Design chapter in the *Innovus Stylus Common UI User Guide*

report_narrow_channel

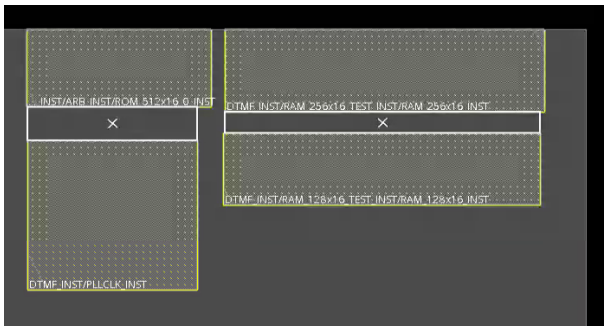
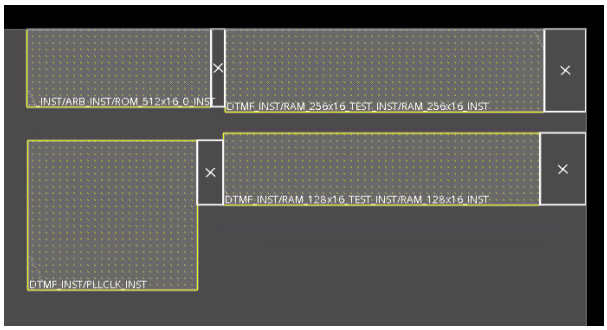
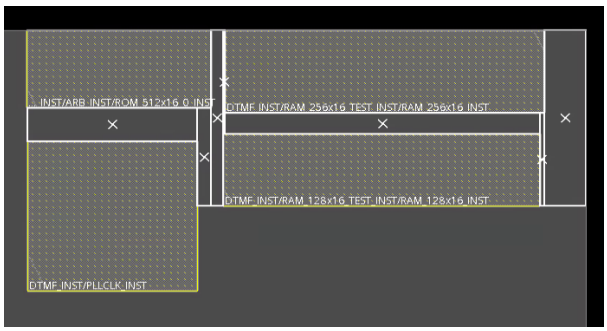
```
report_narrow_channel
[-help]
{-width width
[-ignore_placement_blockage ]
[-no_merge_channel_check ]
[-direction {x y xy}]
[-active_objects {macro macroHalo ioPad ioCell core fence hardBlkg softBlkg partialBlkg routeBlkg row}] }
```

Creates a report that contains a list of narrow channels.

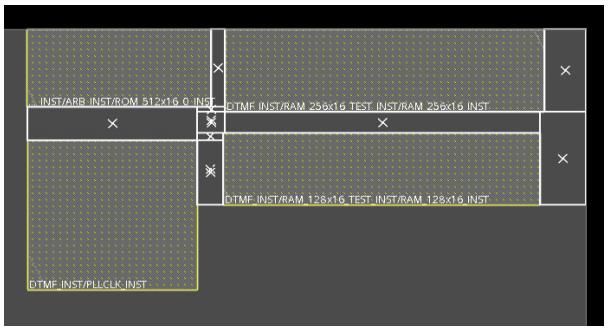
Note: The `report_narrow_channel` command returns channel boxes as its return value.

Parameters

`-active_objects {macro macroHalo ioPad ioCell core fence hardBlkg softBlkg partialBlkg routeBlkg row}`	
	Specifies the active objects for narrow channel checking. *Default*: `"core macro macroHalo hardBlkg"` *Data_type*: (enum)+, optional
`-help`	Outputs a brief description that includes the type and default information for each `report_narrow_channel` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man report_narrow_channel`

`-direction {x y xy}`	Specifies the direction for checking narrow channels. You can specify x, y, or xy.
	Default: `xy`
	Example:
	`report_narrow_channel -width 100 -direction x`
	 
	
	`report_narrow_channel -width 100 -direction y`
	 
	
	`report_narrow_channel -width 100 -direction xy`
	 
	
	Data_type: enum, optional

`-ignore_placement_blockage`	When specified, the placement blockages are ignored while computing channel widths. *Data_type*: bool, optional

`-no_merge_channel_check`	Specifies that the narrow channel error boxes will be merged into contiguous error boxes. Example: `report_narrow_channel -width 100 -direction xy -no_merge_channel_check`  *Data_type*: bool, optional
`-width` *width*	Specifies the width for checking the channels. *Maximum*: `100000` *Minimum*: `0` *Data_type*: float, optional

Example

The following command checks for channels with width 1.0 and ignores placement blockages:
`report_narrow_channel -width 1.0 -ignore_placement_blockage`

Related Information

- Floorplanning the Design chapter in the *Innovus Stylus Common UI User Guide*

report_net_groups

```
report_net_groups
[-help]
[-out_file filename]
```

Creates a report that contains net groups and their nets. You can use this command after importing the design.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `report_net_groups` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man report_net_groups`
`-out_file filename`	Specifies the report filename. If no filename is specified, the report is printed to the console window. *Data_type*: string, optional

Example

Check the bump assignment and send the report to `netGrp.rpt`:

```
report_net_groups -out_file netGrp.rpt
```

The output format is the same as the NetGroup section of the floorplan file:

```
# NetGroup Report File
NetGroup: netGroupName_one
    GroupNet net_name_01_in_netGroupName_one
    GroupNet net_name_02_in_netGroupName_one
NetGroup: netGroupName_two
    GroupNet net_name_01_in_netGroupName_two
    GroupNet net_name_02_in_netGroupName_two
```

Related Information

- `create_net_group`
- `update_net_group`
- `delete_net_group`
- Partitioning the Design chapter in the *Innovus Stylus Common UI User Guide*
- Floorplanning the Design chapter in the *Innovus Stylus Common UI User Guide*

report_obj_connectivity

```
report_obj_connectivity
[-help]
[-direction {in|out|all}]
[-file file_name]
[-level level]
[-no_split]
[-through_registers]
[-to_ports]
[-insts inst+] | [-ports port+] | [-selected]
```

Reports the connections of specified macros, selected macros, or specified ports. You can also report the input and output connectivity separately. By default it reports both.

Parameters

`-direction {in|out|all}`	Specifies the pin direction of objects to be reported. *Default*: `all` *Data_type*: string, optional
`-file file_name`	Specifies the name of the output report file in which the trace macro results will be saved. *Default*: `<top_module_name>.tmcrpt` *Data_type*: string, optional
`-help`	Outputs a brief description that includes type and default information for each `report_obj_connectivity` parameter. For a detailed description of the command and all of its parameters, use the man command: `man report_obj_connectivity`
`-insts inst+`	Specifies the macros to be reported. *Data_type*: string, optional
`-level level`	Specifies the trace level. *Maximum*: 10 *Minimum*: 1 *Default*: 5 *Data_type*: int, optional
`-no_split`	Reports a single object per line.
`-ports port+`	Specifies the IO ports to be reported. *Data_type*: string, optional
`-through_registers`	Reports the through registers between selected objects. *Data_type*: bool, optional
`-to_ports`	Reports the connectivity from the selected objects to IO ports. *Data_type*: bool, optional
`-selected`	Reports the selected macros or IO ports. *Data_type*: bool, optional

Example

The following command reports the selected macros or IO ports:

```
report_obj_connectivity -selected
```

Related Information

- Floorplanning the Design chapter in the *Innovus Stylus Common UI User Guide*

report_off_grid_blocks

```
report_off_grid_blocks
[-help]
```

Reports unsnapped blocks based on the snap preferences (as defined in the *Floorplan Tab* of the *Preferences* form). You can use this command after importing the design.

Parameters

`-help`	Outputs a brief description of the `report_off_grid_blocks` command. For a detailed description of the command, use the man command: `man report_off_grid_blocks`

Related Information

- Floorplanning the Design chapter in the *Innovus Stylus Common UI User Guide*

report_seed_connections

```
report_seed_connections
[-help]
[-constraints_file filename]
[-exclude_nets_file filename]
[-out_file outfile]
```

Reports the number of connections between user-specified seeds.

Parameters

`-constraints_file` *filename*	Specifies the name of the input constraints file containing the seed definition. *Data_type*: string, optional

`-exclude_nets_file` *filename*	Specifies the name of the exclude net file to use during the calculation of seed connections. The exclude_nets_file format should be:
	`#any comment`
	`netName, netName,...`
	`netName, netName,...`
	Data_type: string, optional

`-out_file` *filename*	Specifies the name of the output report file in which the information about the number of connections between user-specified seeds is written. *Data_type*: string, optional
`-help`	Outputs a brief description that includes type and default information for each `report_seed_connections` parameter. For a detailed description of the command and all of its parameters, use the man command: `man report_seed_connections`.

Related Information

- Floorplanning the Design chapter in the *Innovus Stylus Common UI User Guide*

reset_fixed_module_size

```
  reset_fixed_module_size
[-help]
-modules
```

Specifies that during floorplan resizing, the size of the specified modules and/or blackboxes can change. Use this command to unset the fixed block size specified with the `set_module_size_fixed` command. You would use this command after setting the block size of modules and/or black boxes to fixed with the `set_module_size_fixed` command.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `reset_fixed_module_size` parameter. For a detailed description of the command and all of its parameters, use the man command: `man reset_fixed_module_size`
`-modules`	Specifies that the size of the specified modules and/or blackboxes can change during floorplan resizing. Provide a space-separated list of instances. `*`: Specifies that the size of all the modules and/or blackboxes can change during floorplan resizing. *Data_type*: list, required

Example

The following command specifies that the size of instanceA and instanceB can change during floorplan resizing.

```
reset_fixed_module_size instanceA instanceB
```

Related Information

- `set_module_size_fixed`
- Floorplanning the Design chapter in the *Innovus Stylus Common UI User Guide*

reset_macro_place_constraint

```
reset_macro_place_constraint
[-help]
[-align_group]
[-array]
[-avoid_abut_macro_edge_with_pins]
[-cell_obs]
[-cpg]
[-forbidden_space_to_core]
[-forbidden_space_to_macro]
[-honor_strict_spacing_constraint]
[-horizontal_stacking]
[-instance_orientation]
[-macro_corner_keepout]
[-max_io_pin_group_keep_out]
[-min_space_to_core]
[-min_space_to_macro]
[-parallel_run_length]
[-pg_resource_model]
[-same_length_site]
[-track_adjustment]
[-vertical_stacking]
```

Resets the macro constraints that are honored by the macro placer. To define the constraints of macros, use the `set_macro_place_constraint` command.

Parameters

`-align_group`	Deletes all the align_group constraints. *Data_type*: bool, optional
`-array`	Deletes all the array constraints. *Data_type*: bool, optional
`-avoid_abut_macro_edge_with_pins`	
	Specifies that macro edges with pins should not be abutted even when the macro distance is within the forbidden spacing setting. *Data_type*: bool, optional
`-cell_obs`	Deletes all the cell obstruction constraints. *Data_type*: bool, optional
`-cpg`	Deletes all the cpg constraints. *Data_type*: bool, optional
`-forbidden_space_to_core`	Resets the forbidden space to core. *Data_type*: bool, optional
`-forbidden_space_to_macro`	Resets the forbidden space to macro. *Data_type*: bool, optional
`-help`	Outputs a brief description that includes the type and default information for each `reset_macro_place_constraint` parameter. For a detailed description of the command and all of its parameters, use the man command: `man reset_macro_place_constraint`
`-honor_strict_spacing_constraint`	Specifies that forbidden spacing and minimum spacing constraints should be honored strictly. *Data_type*: bool, optional
`-horizontal_stacking`	Resets the maximum horizontal stack depth and the minimum horizontal spacing. *Data_type*: bool, optional

`-instance_orientation`	Deletes all the instance orientations. *Data_type*: bool, optional
`-macro_corner_keepout`	Resets the macro corner keepout constraints. *Data_type*: bool, optional
`-max_io_pin_group_keep_out`	Deletes the macro-only blockages which are automatically added to IO pins and resets the maximum height or width of macro-only blockages to 0. *Data_type*: bool, optional
`-min_space_to_core`	Resets the minimum space to core. *Data_type*: bool, optional
`-min_space_to_macro`	Resets the minimum space to macro. *Data_type*: bool, optional
`-parallel_run_length`	Resets the parallel run length to enable space checking. *Data_type*: bool, optional
`-pg_resource_model`	Deletes the pg resource model constraints. *Data_type*: bool, optional
`-same_length_site`	Resets the minimum difference site number. *Data_type*: bool, optional
`-track_adjustment`	Deletes all the track adjustment. *Data_type*: bool, optional
`-vertical_stacking`	Resets the maximum vertical stack depth and the minimum vertical spacing. *Data_type*: bool, optional

Example

The following command deletes all the pg resource model constraints:

```
reset_macro_place_constraint -pg_resource_model
```

Related Information

- `check_macro_place_constraint`
- `create_pg_model_for_macro_place`
- `get_macro_place_constraint`
- `set_macro_place_constraint`
- `write_macro_place_constraint`
- Floorplanning the Design chapter in the *Innovus Stylus Common UI User Guide*
- Using the Mixed Placer chapter in the *Innovus Stylus Common UI User Guide*

resize_floorplan

```
resize_floorplan
[-help]
[-x_size x_axis_size]
[-y_size y_axis_size]
[-x_ratio x_axis_ratio]
[-y_ratio y_axis_ratio]
[-undo]
[-push_core]
[-force_resize]
```

Resizes the floorplan while maintaining the relative locations of the existing floorplan. You can also set global parameters for the `resize_floorplan` command by using the resize_floorplan Category Attributes. The attributes that you specify are then used automatically whenever you run the `resize_floorplan` command or use the *Resize Floorplan* form. You can use this command after importing the design. When run on a block design, the `resize_floorplan` command keeps physical PG pins during resize and trims the PG pin geometries that fall out of the chip boundary after resize.

Note: You can use the `create_resize_blockage` command to add a size blockage object that controls the behavior of the resize line under a covered area.

Parameters

`-force_resize`	Resizes the floorplan even if the target size can not be met. However, the floorplan is resized as near as possible to the specified size. *Default*: If you do not specify this parameter, the floorplan is not resized if the target size can not be met. **Note**: This parameter can be used only when the `resize_floorplan_shift_based` attribute has been set to `true`. *Data_type*: bool, optional
`-help`	Outputs a brief description that includes type and default information for each `resize_floorplan` parameter. For a detailed description of the command and all of its parameters, use the man command: `man resize_floorplan`
`-push_core`	Pushes the core when the design is I/O bound. *Data_type*: bool, optional
`-undo`	Revert to the floorplan that existed before the command was run. *Data_type*: bool, optional
`-x_ratio x_axis_ratio`	Specifies the percentage of how much the floorplan should be expanded or shrunk in the x direction. A positive value expands the floorplan and a negative value shrinks the floorplan. *Data_type*: float, optional
`-y_ratio y_axis_ratio`	Specifies the percentage of how much the floorplan should be expanded or shrunk in the y direction. A positive value expands the floorplan and a negative value shrinks the floorplan. *Data_type*: float, optional
`-x_size x_axis_size`	Specifies how much the floorplan should be expanded or shrunk in the x direction. A positive value expands the floorplan and a negative value shrinks the floorplan. This value is in microns. **Note**: If both `-x_size` and `-y_size` parameters are specified, the resize order is the same as the order of the parameters. *Data_type*: float, optional
`-y_size y_axis_size`	Specifies how much the floorplan should be expanded or shrunk in the y direction. A positive value expands the floorplan and a negative value shrinks the floorplan. This value is in microns. *Data_type*: float, optional

Example

The following command resizes the floorplan as follows: shrinks the floorplan in shift based mode by -50 microns in the x direction and -40 microns in the y direction.

```
resize_floorplan -x_size -50 -y_size -40
```

Related Information

- `create_resize_blockage`
- `resize_floorplan` Category Attributes
- `set_resize_line`
- Floorplan Menu chapter in the *Innovus Stylus Common UI Menu Reference*
- Floorplanning the Design chapter in the *Innovus Stylus Common UI User Guide*
  - Resizing the Floorplan

resize_floorplan_channel

```
resize_floorplan_channel
[-help]
[-channel_density value]
[-constraints constraint_file]
[-move_only]
[-report output_file]
[-report_only [-report_low_density_channels value]
  [-types {all fence_to_fence fen_to_core fence_to_macro macro_to_macro macro_to_core}]]
```

Automatically adjusts the channel width between objects to prevent potential routing congestion. You can use this command after importing the design, loading a completed floorplan, and then running the `place_design` and `route_early_global` commands.

Parameters

`-channel_density value`	
	Specifies the maximum utilization of routing tracks in a channel. *Data_type*: float, optional
`-constraints constraint_file`	
	Specifies the file that contains constraints for floorplan objects. A constraint file is a text file that can contain a list of seeds to use when generating fences and module guides, and basic relative, spacing and orientation constraints to follow during macro placement. *Data_type*: string, optional
`-help`	Outputs a brief description that includes type and default information for each `resize_floorplan_channel` parameter. For a detailed description of the command and all of its parameters, use the man command: `man resize_floorplan_channel`
`-move_only`	Specifies that only move-only adjustment is allowed. *Data_type*: bool, optional
`-report output_file`	Generates a report that contains the channel congestion, targeted channel adjustment, and final real floorplan adjustment. *Default*: `topcell.channel` *Data_type*: string, optional
`-report_low_density_channels value`	
	Reports channel congestion whose track utilization is lower than the user-specified threshold value. *Minimum*: 0 *Maximum*: 1.0 *Data_type*: float, optional
`-report_only`	Generates a report of the channel congestion without adjusting the floorplan. *Data_type*: bool, optional
`-types {all fenceToFence fenceToCore fenceToMacro macroToMacro macroToCore}`	
	Reports channel congestion for specified type. You can specify more than one type at a time. *Default*: `all` *Data_type*: (enum)+, optional

Example

- The following command generates a report of only fence-to-fence congestion, without adjusting the floorplan:
  ```
  resize_floorplan_channel -report_only -types {fenceToFence}
  ```
- The following command generates a channel congestion report without adjusting the floorplan:
  ```
  resize_floorplan_channel -report_only -report reportfile
  ```

  The following report is generated:

  ```
  Channel (315.48 357.67 361.60 511.57)
  Left: CORE
  Right: DTMF_INST/PLLCLK_INST
  Congested utilization: 200/185 (108.1% V)
  Expected channel width increase: 6.48 um

  Channel (315.48 511.57 351.33 646.17)
  Left: CORE
  Right: DTMF_INST/RAM_128x16_TEST_INST/RAM_128x16_INST
  Congested utilization: 128/117 (109.4% V)
  Expected channel width increase: 5.51 um

  Channel (466.52 646.17 792.28 670.01)
  Bottom: DTMF_INST/RAM_128x16_TEST_INST/RAM_128x16_INST
  Top: DTMF_INST/ROM_512x16_0_INST
  Congested utilization: 52/42 (123.8% H)
  Expected channel height increase: 7.38 um
  ```

Related Information

- `place_design`
- `route_early_global`
- `resize_floorplan`
- Floorplanning the Design chapter in the *Innovus Stylus Common UI User Guide*

resize_floorplan Category Attributes

The root attributes in the `resize_floorpan` category are:

- resize_floorplan_cong_aware
- resize_floorplan_honor_halo
- resize_floorplan_io_fix
- resize_floorplan_io_move_with_edge
- resize_floorplan_io_relative
- resize_floorplan_maintain_resource_ratio
- resize_floorplan_relative
- resize_floorplan_shift_based
- resize_floorplan_shrink_fence
- resize_floorplan_snap_to_track

To set the value of any root attribute, use the `set_db` command:

```
set_db attribute_name value
```

To get the current value of any root attribute, use the `get_db` command:

```
get_db attribute_name
```

To get the current value of all attributes in the resize_floorpan category, use the following command:

```
get_db -category resize_floorpan
```

resize_floorplan_cong_aware

```
resize_floorplan_cong_aware {true | false}
```

Default: `false`

Data_type: bool, read/write

Resizes and shifts the floorplan objects by estimating the congestion for the floorplan and automatically deciding where to draw a resize line to avoid the congested area.

Related Commands

- `create_resize_blockage`
- `resize_floorplan`

resize_floorplan_honor_halo

```
resize_floorplan_honor_halo {true | false}
```

Default: `true`

Data_type: bool, read/write

Honors placement halo and preserves the space between macros held by halo after the floorplan resize. With this option set to true, if the original placement halo does not overlap with other macros or placement halo, after floorplan resize, it will not overlap with other macros or placement halo.

Related Commands

- `create_resize_blockage`
- `resize_floorplan`

resize_floorplan_io_fix

```
resize_floorplan_io_fix {true | false}
```

Default: `false`

Data_type: bool, read/write

Fixes I/Os at current location.

Related Commands

- `create_resize_blockage`
- `resize_floorplan`

resize_floorplan_io_move_with_edge

```
resize_floorplan_io_move_with_edge {true | false}
```

Default: `false`

Data_type: bool, read/write

Moves pins on the moved edge orthogonally to edge direction so pins stay on edge after movement. The pins on other edges are not touched.

Related Commands

- `create_resize_blockage`
- `resize_floorplan`

resize_floorplan_io_relative

```
resize_floorplan_io_relative {true | false}
```

Default: `true`

Data_type: bool, read/write

Resizes the space among I/Os proportionally. If you set this parameter to true, the I/Os are distributed proportional to the original I/O space. If you set this parameter to false, the I/Os are distributed evenly.

Note: By default, the overlapping I/Os are spaced proportionally.

Related Commands

- `create_resize_blockage`
- `resize_floorplan`

resize_floorplan_maintain_resource_ratio

```
resize_floorplan_maintain_resource_ratio {true | false}
```

Default: `false`

Data_type: bool, read/write

When enabled, it maintains the resource ratio after resizing the floorplan in shift-based mode.

Related Commands

- `create_resize_blockage`
- `resize_floorplan`

resize_floorplan_relative

```
resize_floorplan_relative {true | false}
```

Default: `false`

Data_type: bool, read/write

Resizes the space among floorplan objects proportionally.

Related Commands

- create_resize_blockage
- resize_floorplan

resize_floorplan_shift_based

```
resize_floorplan_shift_based {true | false}
```

Default: `true`

Data_type: bool, read/write

Shifts floorplan objects at appropriate location(s) without changing the proportional spacing. You can specify resize lines to control the areas of floorplan that will be expanded or shrunk. To set resize lines, use the `set_resize_line` command. If you do not specify resize line(s), the floorplan is resized automatically - that is, the resize lines are derived automatically based on the current floorplan and the specified resize values.

Related Commands

- create_resize_blockage
- resize_floorplan
- set_resize_line

resize_floorplan_shrink_fence

```
resize_floorplan_shrink_fence {true | false}
```

Default: `false`

Data_type: bool, read/write

Enables resized lines to go through fences and regions in floorplan. With this option enabled, during floorplan resize,the fences, regions, and guides inside the floorplan are automatically resized while maintaining their minimum size after the shrink.

Related Commands

- create_resize_blockage
- resize_floorplan

resize_floorplan_snap_to_track

```
resize_floorplan_snap_to_track {true | false}
```

Default: `false`

Data_type: bool, read/write

Snaps resize values (shrink/expand) of the floorplan to a multiple integer of the metal layer pitch. For example, if the horizontal metal pitch is 1.568 microns and you want to shrink the floorplan by 8 microns in y direction, the actual shrink value is 7.84 microns, the nearest multiple integer of the metal pitch. The x direction resize value snaps to the first vertical layer pitch and the y direction resize value snaps to the first horizontal layer pitch.

Related Commands

- create_resize_blockage
- resize_floorplan

resize_partition_to_abut

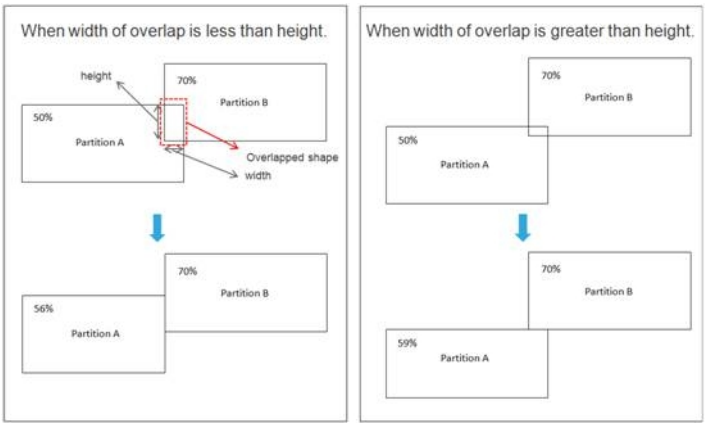
```
resize_partition_to_abut
[-help]
[-cut_to_fix_overlap]
[-honor_place_blockage blockage_name]
[-partition_to_core_gap distance]
[-partition_to_partition_gap distance]
{-hinsts hierarchical_instance_list | -selected | -all}
```

Creates an abutted floorplan from an existing floorplan by fixing gaps between:

- Partitions
- Partition and core boundary
- Partition and macro when there is no overlap between the partition and the macro

Note:

- The `resize_partition_to_abut` command honors place blockage when the blockage has no overlap with the partition. It does not honor doughnut and master/clone partitions.
- When you use the `resize_partition_to_abut` command, partitions grow. The order in which partitions grow is top edge,bottom edge,left edge,right edge. Also, some partitions shrink when the `-partiton_to_partiton_gap` option is used. The order in which partitions shrink is left edge, top edge, right edge, bottom edge.
- When there is an overlap between two partitions, if the width of the overlapped shape is less than the height, the partition with the lower TU shrinks horizontally. However, if the width of overlapped shape is bigger than the height, the partition with the lower TU shrinks vertically.

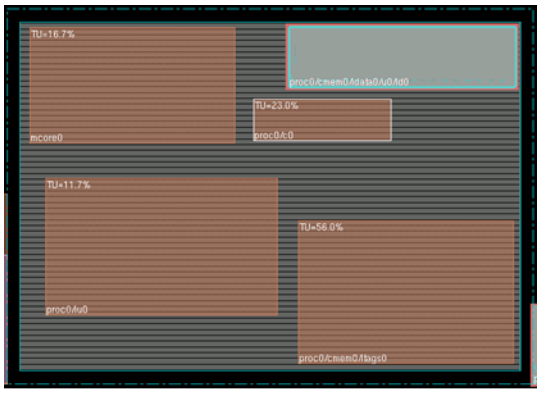


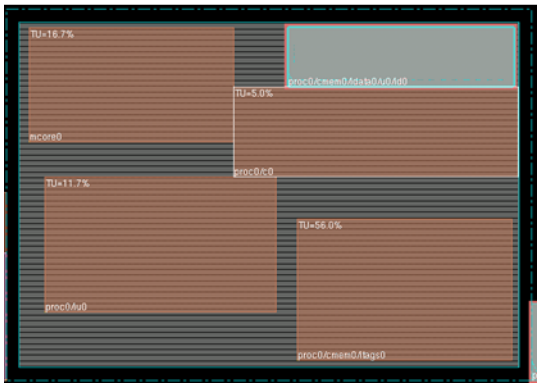
Parameters

`-help`	Outputs a brief description that includes the type and default information for each `resize_partition_to_abut` parameter. For a detailed description of the command and all of its parameters, use the man command: `man resize_partition_to_abut`
`-all`	Specifies to grow all partitions in the design. *Data_type*: bool, optional
`-cut_to_fix_overlap`	Cuts the overlapped partitions, if specified. Otherwise, the tool will shrink the partitions instead of cutting. *Data_type*: bool, optional
`-hinsts` *hierarchical_instance_list*	Specifies a list of hierarchical instances that will be modified. *Default*: All partitions in the design. *Data_type*: string, optional
`-honor_place_blockage` *blockage_name*	Specifies to fill gap upto the placement blockage boundary only. If a blockage name is specified, it honors only the specified blockage name. *Default*: Does not honor placement blockage. *Data_type*: string, optional
`-partition_to_core_gap` *distance*	Specifies the required distance (in micron) between a partition boundary and and a core boundary after growing the partition. *Default*: 0 micron *Data_type*: float, optional
`-partition_to_partition_gap` *distance*	Specifies the required distance (in micron) between the partitions after growing the partitions. *Default*: 0 micron *Data_type*: float, optional
`-selected`	Specifies to grow selected partitions. *Data_type*: bool, optional

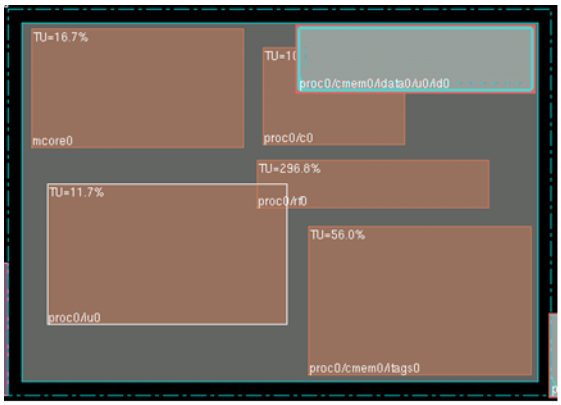
Examples

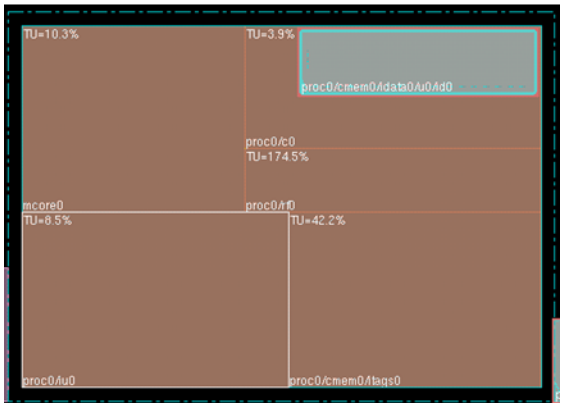
- The following command grows the selected partition
  ```
  resize_partition_to_abut -selected
  ```





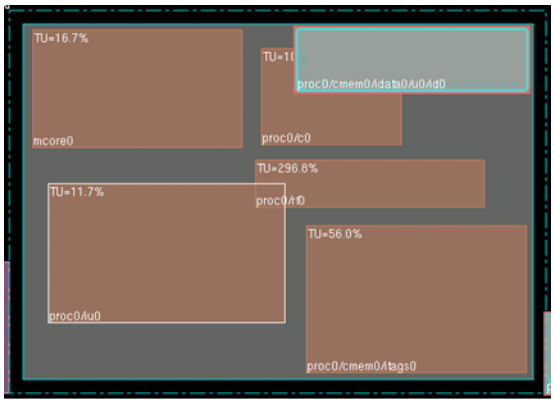
- If a partition overlaps another partition, the partition with higher target utilization keeps the original shape. The other partition shrinks to remove the overlap.
  The following command keeps the originals shape of proc0/rf0 and shrinks the other partitions to remove the overlap.
  ```
  resize_partition_to_abut -all
  ```

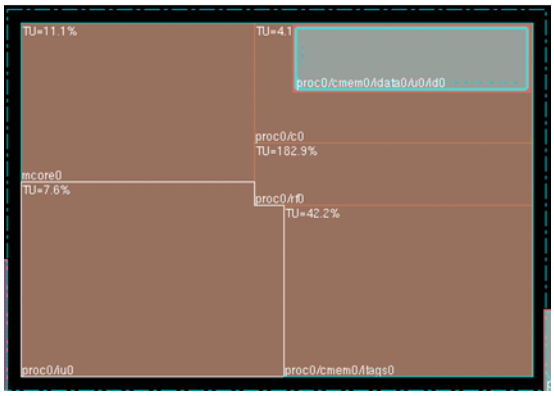




- The following command cuts the proc0/iu0 partition and keeps the original shape of the proc0/rf0 partition, which has higher utilization, to remove the overlap.

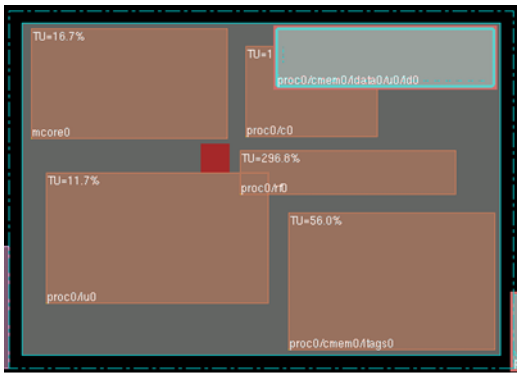
```
resize_partition_to_abut -cut_to_fix_overlap -all
```

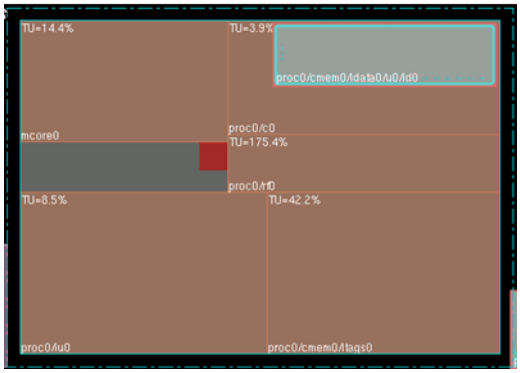




- The following command honors the place blockage which is not overlapping the partition by default.

```
resize_partition_to_abut -all -honor_place_blockage all
```





- The following command keeps the gap between two partitions, and between partitions and the core boundary after growing partition.

```
resize_partition_to_abut -partition_to_core_gap 5 -partition_to_partition_gap 5 -all -cut_to_fix_overlap
```

Related Information

- Floorplanning the Design chapter in the *Innovus Stylus Common UI User Guide*

set_all_ios_to_fixed

```
set_all_ios_to_fixed
[-help]
[-pin_only | -cell_only | -area_io]
```

Changes the status of all I/O pins, I/O cells, or CLASS PAD AREAIO cells to a FIXED state to keep them from being reassigned. Use this command after importing the design.

Parameters

`-cell_only`	Changes the status of all I/O cells to a FIXED state. *Data_type*: bool, optional
`-help`	Outputs a brief description that includes the type and default information for each `set_all_ios_to_fixed` parameter. For a detailed description of the command and all of its parameters, use the man command: `man set_all_ios_to_fixed`
`-area_io`	Changes the status of all CLASS PAD AREAIO cells to a FIXED state. *Data_type*: bool, optional
`-pin_only`	Changes the status of all I/O pins to a FIXED state. *Data_type*: bool, optional

Examples

- The following command changes the status of all I/O pins and I/O cells to a FIXED state:
  ```
  set_all_ios_to_fixed
  ```
  Note: This is the default functionality.
- The following command changes the status of all I/O pins to a FIXED state:
  ```
  set_all_ios_to_fixed -pin_only
  ```
- The following command changes the status of all I/O cells to a FIXED state:
  ```
  set_all_ios_to_fixed -cell_only
  ```
- The following command changes the status of all I/O pins, I/O cells, and all CLASS PAD AREAIO cells to a FIXED state:
  ```
  set_all_ios_to_fixed -area_io
  ```

Related Information

- Floorplanning the Design chapter in the *Innovus Stylus Common UI User Guide*

set_bottom_io_pad_orient

```
set_bottom_io_pad_orient
[-help]
orient
```

Changes the orientation for the bottom pad (South side). Changing the bottom pad orientation changes the pad relationship of the East, North, and West side pads. You can use this command after importing the design.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `set_bottom_io_pad_orient` parameter. For a detailed description of the command and all of its parameters, use the man command: `man set_bottom_io_pad_orient`
orient	Specifies the orientation. The possible orientation values are `r0`, `r90`, `r180`, and `r270`. *Data_type*: string, required

Example

The following command changes the orientation of the bottom I/O pad instances from r0 to r270 for the bottom pad (South side):

```
set_bottom_io_pad_orient r270
```

As a result, the respective orientation of the I/O pad instances change on the East (r270 to r180), North (r180 to r90), and West (r90 to r0) sides.

Related Information

- Floorplanning the Design chapter in the *Innovus Stylus Common UI User Guide*

set_flipping

```
set_flipping
[-help]
flip_type
```

Specifies the orientation of the bottom row in the core area. The command recreates all the floorplan rows. You can use this command after importing the design.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `set_flipping` parameter. For a detailed description of the command and all of its parameters, use the man command: `man set_flipping`
`flip_type`	Specifies the type of orientation. • `f`: Specifies that the first row flips from the bottom up. • `n`: Specifies no row flipping. • `s`: Specifies that the second row flips from the bottom up. **Tip:** To get the flipping value set by the `set_flipping` command, use the following: `get_db designs .row_flip` *Data_type*: string, required

Examples

- If the existing orientation of the bottom row is Mx and the second row from bottom is R0, then the following command changes the second row orientation to Mx and changes the orientation of the other rows accordingly (bottom row orientation will be R0).
  ```
  set_flipping s
  ```
- If the existing orientation of the bottom row is R0 and second row from bottom is Mx, then the following command changes the first row orientation to Mx and changes the orientation of the other rows accordingly (second from bottom row orientation will be R0):
  ```
  set_flipping f
  ```

Related Information

- Floorplanning the Design chapter in the *Innovus Stylus Common UI User Guide*

set_floorplan_row_spacing_and_type

```
set_floorplan_row_spacing_and_type
[-help]
spacing
type
```

Specifies the standard row spacing values and row pattern type. You can use this command after importing the design.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `set_floorplan_row_spacing_and_type` parameter. For a detailed description of the command and all of its parameters, use the man command: `man set_floorplan_row_spacing_and_type`
`spacing`	Specifies the standard row spacing, in micrometers. This must be a positive value. *Default*: 0 *Data_type*: float, required
`type`	Specifies the row spacing type. Specify 1 for every row, or 2 for every other row to apply the spacing value. *Data_type*: int, required

Example

The following command specifies a row spacing value of 1.1 micrometers for every other row:

```
set_floorplan_row_spacing_and_type 1.1 2
```

Related Information

- Floorplanning the Design chapter in the *Innovus Stylus Common UI User Guide*
  - Standard Row Spacing

Data_type: float, required

set_inst_group_physical_hier

```
set_inst_group_physical_hier
[-help]
inst_group
```

Sets a pre-existing instance group into a physical hierarchy where you can generate a corresponding netlist. You can use this command after creating a group name.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `set_inst_group_physical_hier` parameter. For a detailed description of the command and all of its parameters, use the man command: `man set_inst_group_physical_hier`
`inst_group`	Specifies the name of the created group. *Data_type*: string, required

Example

The following commands create and set the instance group `adder1` as a physical hierarchy:

```
createInstGroup adder1
set_inst_group_physical_hier adder1
```

Related Information

- `create_group`
- Floorplanning the Design chapter in the *Innovus Stylus Common UI User Guide*

set_instance_placement_status

```
set_instance_placement_status
[-help]
-status {unplaced | fixed | placed | cover | softfixed}
{-all_hard_macros | -all_partition_blocks |-all_black_boxes | -name instName(s)}
```

Changes the placement attributes status for all hard macros, all partition blocks, all blackboxes, or selected instances. This allows you to fix all blocks so they are not moved by `place_design`.

Parameters

`-all_hard_macros`	Applies the status option only to all hard macros. *Data_type*: bool, required
`-all_partition_blocks`	Applies the status option only to all partition blocks. *Data_type*: bool, required
`-all_black_boxes`	Applies the status option only to all blackboxes. *Data_type*: bool, required
`-help`	Outputs a brief description that includes type and default information for each `set_instance_placement_status` parameter. For a detailed description of the command and all of its parameters, use the man command: `man set_instance_placement_status`
`-name instName(s)`	Applies the status options to the specified instance(s). You can use a wildcard to specify multiple instances. *Data_type*: string, required
`-status {unplaced | fixed | placed | cover | softfixed}`	
	Specifies the change in placement status for instances. • `unplaced`: Specifies the change in placement status to be an *unplaced* status. The unplaced component does not have a location. • `fixed`: Specifies the change in placement status to a *fixed* status. The fixed component has a location and cannot be moved by automatic tools, but can be moved using interactive commands. • `placed`: Specifies the change in placement status to a *placed* status. The placed component has a location, and can be moved using automatic tools and interactive commands. • `cover`: Specifies the change in placement status to a *cover* status. The cover component has a location, but cannot be moved by automatic tools or interactive commands. • `softfixed`: Specifies the change in placement status to a *softFixed* status. The softFixed placement status means that instances cannot be moved by global placement and can only be moved by the legalization step of detail placement. Instances with this status can also be upsized by optimization. *Data_type*: enum, required

Example

The following commands set all hard macros and blackboxes to `fixed`. By setting this status, `place_design` can run without moving these fixed blocks.

```
set_instance_placement_status -all_hard_macros -status fixed
set_instance_placement_status -all_black_boxes -status fixed
```

Related Information

- `place_design`
- Floorplanning the Design chapter in the *Innovus Stylus Common UI User Guide*

set_io_constraints

```
set_io_constraints
[-help]
[-force_non_io_row]
[-io_order {clockwise | counterclockwise | default} | -row string]
[-spacing num | -remove_spacing]
[-end_space num | -remove_end_space]
[-first_cell_spacing float| -remove_first_cell_spacing]
[-remove_all]
```

Changes the constraints of an I/O row defined in the I/O constraint file. If you do not specify the name of the I/O row, the command modifies all the I/O rows. You can use this command to move a IO pad .

Parameters

`-end_space num`	Specifies the space, in µmeters, between the corner pad and the last I/O pad for the specified side of the row. *Default*: `0` *Data_type*: float, optional
`-first_cell_spacing float`	
	Specifies the distance, in µmeters, from the first cell to the beginning of the I/O row, based on the I/O order. *Data_type*: float, optional
`-force_non_io_row`	Forces the I/O constraints to run on a non I/O row flow. *Data_type*: bool, optional
`-help`	Outputs a brief description that includes type and default information for each `set_io_constraints` parameter. For a detailed description of the command and all of its parameters, use the man command: `man set_io_constraints.`
`-io_order {clockwise | counterclockwise | default}`	
	Changes the I/O placement order on all the rows. **Note:** This parameter is ignored if you specify a row name. *Default*: `default` *Data_type*: string, optional
`-remove_all`	Removes all the existing constraints on the I/O's. Evenly distributes all the I/O's in the rows. *Data_type*: bool, optional
`-remove_end_space`	Removes the end space of the row. *Data_type*: bool, optional
`-remove_first_cell_spacing`	
	Removes the spacing of the first cell. *Data_type*: bool, optional
`-remove_spacing`	Remove the spacing constraint from I/O cells on a row. *Data_type*: bool, optional
`-row string`	Specifies the name of the row to modify the I/O constraints. *Default*: 1 *Data_type*: string, optional
`-spacing num`	Specifies the global spacing, in µmeters, for all the I/O's on a row. **Note:** `set_io_constraints -io_order -spacing num` does not modify the spacing of corner I/O pads. *Default*: 0 *Data_type*: float, optional

Example

- The following command removes the spacing constraint from all I/O rows defined in the I/O constraint file:
  ```
  set_io_constraints  -remove_spacing
  ```

Related Information

- `create_io_row`
- Floorplanning the Design chapter in the *Innovus Stylus Common UI User Guide*

- Performing I/O Row Based Pad Placement

- Data Preparation chapter in the *Innovus Stylus Common UI User Guide*
  - Creating an I/O Assignment File

set_io_flow_flag

```
set_io_flow_flag
[-help]
{0 | 1}
```

Sets the flag to use I/O row flow for pad placement. To view the current flag setting, use the `get_io_flow_flag` command. You can use this command in the floorplanning stage. You can use this commands before creating I/O rows for I/O row based pad placement.

Parameters

`{0 | 1}`	Specify 1, to use the I/O row flow. Specify 0, to use the normal I/O flow. *Data_type*: int, required
`-help`	Outputs a brief description that includes type and default information for each `set_io_flow_flag` parameter. For a detailed description of the command and all of its parameters, use the man command: `man set_io_flow_flag.`

Example

- The following command enables you to start using the I/O row flow in Innovus:
  ```
  set_io_flow_flag  1
  ```

Related Information

- `create_io_row`
- `get_io_flow_flag`
- Floorplan Menu chapter in the *Innovus Stylus Common UI Menu Reference*
  - Create I/O Row
- Floorplanning the Design chapter in the *Innovus Stylus Common UI User Guide*
  - Performing I/O Row Based Pad Placement
- Data Preparation chapter in the *Innovus User Guide*
  - Creating an I/O Assignment Flow

set_io_row_margin

```
set_io_row_margin
[-help]
side
row
margin
```

Sets the distance from the die boundary edge to the I/O row starting edge location. You can use this command for multiple I/O rows.

Note: You can use the *Edit I/O Ring* form to specify I/O pad rings and row margins for multiple rows. Alternatively, to achieve the same using text commands, you must first use the `set_io_row_margin` command to set the distance from the die boundary edge to start of each row and then use the `place_pad_ios` command to place the I/O pads evenly between these rows.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `set_io_row_margin` parameter. For a detailed description of the command and all of its parameters, use the man command: `man set_io_row_margin`
side	Specifies the side of the I/O row margin, where n is the North side, w is the West side, s is the South side, and e is the East side. *Data_type*: string, optional
margin	Specifies the distance between the die boundary edge to the I/O row edge, in microns. *Data_type*: float, optional

row	Specifies the I/O row number. This value can be up to 8 rows. You can also use this value in conjunction with the `place_pad_ios` and `place_peripheral_ios -opt_ios` commands to specify the number of rows for placing peripheral I/O cells for the CLASS PAD and CLASS PAD AREAIO flows respectively.
	• To create and optimize multiple I/O Rows in a CLASS PAD flow, use the following commands:
	`set_io_row_margin row`
	`place_pad_ios`
	• To create and optimize multiple I/O Rows in a CLASS PAD AREAIO flow, use the following commands:
	`set_io_row_margin rowNumber`
	`place_pad_ios -opt_ios`
	Maximum: 1000
	Minimum: 0
	Data_type: int, optional

Example

The following command sets the North side of the second I/O row starting edge value at 300 microns from the die boundary edge to the second I/O row edge:

```
set_io_row_margin n 2 300
```

Related Information

- `place_pad_ios`
- `place_peripheral_ios`
- Floorplanning the Design chapter in the *Innovus Stylus Common UI User Guide*
- Floorplan Menu chapter in the *Innovus Stylus Common UI Menu Reference*
  - Edit I/O Ring

set_macro_place_constraint

```
set_macro_place_constraint
[-help]
[array_elements]
[-array array_name [-common_hier name] [-valid_group_orients {r0 mx my}] [-place_order {horizontal vertical}]]
[{-insts string | -all_macros } {-orientation string} [-common_hier name]]
[-cells string {-cell_obs string | -track_adjustment string}]
[-align_group string] [-cpg string]
[[-max_io_pin_group_keep_out {depth_value [side_value]}]
[-forbidden_space_to_core {unified_value | horizontal_value vertical_value}]
[-forbidden_space_to_macro {unified_value | horizontal_value vertical_value}]
[-min_space_to_core {unified_value | horizontal_value vertical_value}]
[-min_space_to_macro {unified_value | horizontal_value vertical_value}]
[-parallel_run_length float]
[-horizontal_stacking {max_macro_stack_length min_space_between_macro_stack max_space_between_macros}]
[-vertical_stacking {max_macro_stack_length min_space_between_macro_stack max_space_between_macros}]
[-honor_strict_spacing_constraint {true | false}]
[-avoid_abut_macro_edge_with_pins {true | false}]
[-same_length_site integer]
[-pg_resource_model {M1 value1 M2 value2 …}]
[-macro_corner_keepout {double_endcap_width double_endcap_height}]
[-halo_sharing {true | false}]]
```

Enables you to specify the constraints for placing the macros and standard cells concurrently. These constraints can be used to place macros in an array (or matrix) and define the relative placement of each macro. The whole array, treated like one macro, can be flipped or mirrored (R0, MX, MY, R180). You can use the new `set_macro_place_constraint` command to specify:

- Macro array constraints
- Group aligning constraints
- Macro-to-macro and macro-to-core boundary spacing constraints
- Macro orientation constraints
- Over macro power routing modeling
- Maximum stacking length
- Reserved space for pin routing
- I/O Pin Keep-out

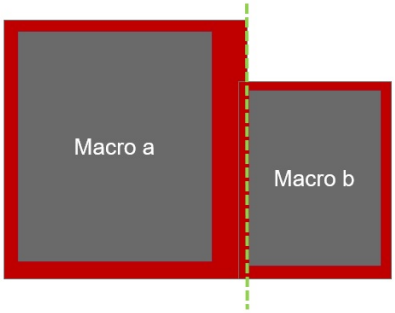
Parameters

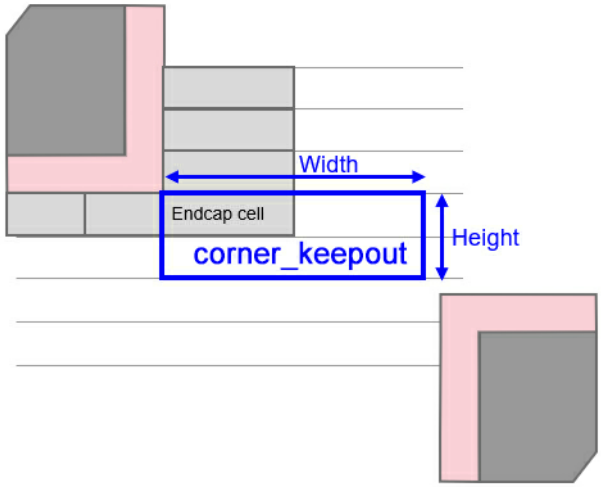
`-align_group string`	Specifies the macros that are to be aligned. *Data_type*: string, optional
`-all_macros`	Specifies that all macros should honor the constraints. *Data_type*: bool, optional
`array_elements`	Specifies the instance members information of a row or column of the array in the following format: *inst_name*:*orientation* [*spacing inst_name*:*orientation*] + **Note:** The `array_elements` should be specified with the `-array` parameter. *Data_type*: string, optional
`-array array_name`	Specifies the macro array name. *Data_type*: string, optional

`-avoid_abut_macro_edge_with_pins {true | false}`	When set to true, the `place_macro_detail` command avoids macro edge abutment even if the macro distance is within the forbidden space setting. When set to true, the `place_macro_detail` command abuts macros regardless of pins. *Default*: `true` *Data_type*: enum, optional
`-cell_obs {{Layer1_Name PowerStripeWidth SpacingBetweenStripes} {Layer2_Name ….}}`	

	Specifies the layer width and spacing to add cell obstruction on each macro to model power stripes . During the concurrent macro placement the power mesh does not exist. This parameter allows to mimic the number of routing tracks occupied by the real power mesh over the macros to make the congestion analysis more accurate during the concurrent macro placement.
	The format is:
	`cell_obs{{Layer1_Name PowerStripeWidth SpacingBetweenStripes} {Layer2_Name ….}}`
	Example:
	`set_macro_place_constraint -cells <cell_name> -cell_obs {{M5 0.24 1.251} {M8 0.64 8.409} {M9 1.2 7.49} }`
	Note: This setting is generated automatically by the `create_pg_model_for_macro_place` command. It is highly recommended to use this command to define the `-cell_obs` parameter.
	Data_type: string, optional

`-cells string`	Specifies the constraints for all instances of the specified cells. *Data_type*: string, optional
`-common_hier name`	Specifies the common logic hierarchy name for *inst_name* in *array_elements*, so the *inst_name* can be specified with short-hand format. *Data_type*: string, optional
`-cpg string`	Specifies the cpg guide of one instance. *Data_type*: string, optional
`-forbidden_space_to_core {unified_value | horizontal_value vertical_value}`	
	Specifies the value for forbidden spacing between macro to core boundary. If the space between the macro and core is smaller than this value, then the macros are abutted. Note: • If only one unified value is specified, both horizontal and vertical spacings use the same value. If two values are specified, the first value is for the horizontal spacing, the second value is for the vertical spacing. • In case the `-min_space_to_core` parameter has been specified, then the value specified for the `-forbidden_space_to_core` parameter should be less than the value specified for the `-min_space_to_core` parameter. *Minimum=0* *Maximum=100000* *Data_type:* string, optional
`-forbidden_space_to_macro {unified_value | horizontal_value vertical_value}`	

	Specifies the value for forbidden spacing between a macro to macro. If the space between the macros is smaller than this value, then the macros are abutted. **Note:** • If only one unified value is specified, both horizontal and vertical spacings use the same value. If two values are specified, the first value is for the horizontal spacing, the second value is for the vertical spacing. • In case the `-min_space_to_macro` parameter has been specified, then the value specified for the `-forbidden_space_to_macro` parameter should be less than the value specified for the `-min_space_to_macro` parameter. *Minimum=0* *Maximum=100000* *Data_type*: string, optional
`-halo_sharing {true | false}`	When enabled, abutted macros can share halos between them. **Note:** When the spacing between two macros is less than the forbidden-space-to-macro, the two macros are abutted. Since a single halo width instead of double width is enough to meet the physical rules, abutted macros can share halos between them using the new `-halo_sharing` parameter. **Note:** The halos of the abutted macros can overlap but macros cannot overlap with the halo of the other macro.  *Default:* `false` *Data_type*: enum, optional
`-help`	Outputs a brief description that includes the type and default information for each `set_macro_place_constraint` parameter. For a detailed description of the command and all of its parameters, use the man command: `man set_macro_place_constraint`
`-honor_strict_spacing_constraint {true | false}`	
	When set to `true`, the `place_macro_detail` command strictly honors the forbidden spacing and minimum spacing constraints. When set to `false`, the `place_macro_detail` command treats forbidden spacing and min spacing setting as soft constraints to avoid large macro movements. *Default*: `false` *Data_type*: enum, optional
`-horizontal_stacking {max_macro_stack_length min_space_between_macro_stack max_space_between_macros}`	

	Specifies the maximum horizontal stack depth and the minimum horizontal spacing. • *max_macro_stack_length:* Specifies that the macro stacking length should not be more than the specified value. • *min_space_between_macro_stack:* Specifies the spacing between different macro stacks. • *max_space_between_macros:* Specifies the maximum space between macros. When the macro spacing is less than the specified value, the macros are counted in the stacking length. For example, if the max_space_between_macro value is 9 then the macros will be considered as packed if the space between the macros is between 0 and 9, however, if the real space is 10 then nothing will happened. *Data_type*: string, optional
`-insts string`	Specifies the macros on which the constraints are set. *Data_type*: string, optional
`-macro_corner_keepout {double_endcap_width double_endcap_height}`	
	Specifies the macro corner spacing constraints. It ensures that there is enough space between the macro corners for inserting the endcap cells properly. **Note:** The corner keepout area starts at the corner of the macro halo.  **Note:** The macro corner spacing constraints specified with this parameter are honored by the `place_macro_detail` and `check_macro_place_constraint` commands. However, if honoring the corner keepout constraint is difficult, corner abutting is allowed. **Note:** The tool reserves either the horizontal keepout larger than double corner cell width or vertical keepout larger than double corner cell height *Default*: {0,0} *Data_type*: string, optional

`-max_io_pin_group_keep_out {depth_value [side_value]}`	Specifies the maximum height and (optionally) side extension to automatically add macro-only blockages to IO pins.
	Note: The `-max_io_pin_group_keep_out` parameter automatically coordinates the size of the macro-only blockages based on the pin density and number for each group of IO pins.
	Note: Side value is optional. If you do not specify the side value, the `set_macro_place_constraint` command follows the default method to calculate the side (side = ¼ depth). If you specify the side value, the max IO group needs to follow the specified depth and side distance. The blockages of smaller IO groups will be derived automatically, keeping the same *depth/side* proportion (as described for the max IO group)
	Innovus supports the following formats:
	`set_macro_place_constraint -max_io_pin_group_keep_out depth_value`
	`set_macro_place_constraint -max_io_pin_group_keep_out {depth_value}`
	`set_macro_place_constraint -max_io_pin_group_keep_out {depth_value side_value}`
	Note: The depth and side values are specified in microns.
	Minimum=0
	Maximum=100000
	Data_type: string, optional

`-min_space_to_core {unified_value | horizontal_value vertical_value}`	
	Specifies the value for the minimum spacing between macro and core boundary. If the space between the macro and core is smaller than this value and larger than the forbidden space, then the macros are pushed away from the core to minimum space. **Note:** If only one unified value is specified, both horizontal and vertical spacings use the same value. If two values are specified, the first value is for the horizontal spacing, the second value is for the vertical spacing. *Minimum=0* *Maximum=100000* *Default:* `0` *Data_type*: string, optional
`-min_space_to_macro {unified_value | horizontal_value vertical_value}`	

	Specifies the value for the minimum spacing between a macro to macro. If the space between the macros is less than this value, then the macros are aligned.
	Note: If only one unified value is specified, both horizontal and vertical spacings use the same value. If two values are specified, the first value is for the horizontal spacing, the second value is for the vertical spacing.
	For Example:
	• In the following command, both horizontal and vertical spacing use the same value.
	`set_macro_place_constraint -min_space_to_macro 10`
	• In the following command, the first value is for horizontal spacing, the second value is for vertical spacing.
	`set_macro_place_constraint -min_space_to_macro {10 20}`
	Minimum=0
	Maximum=100000
	Data_type: string, optional

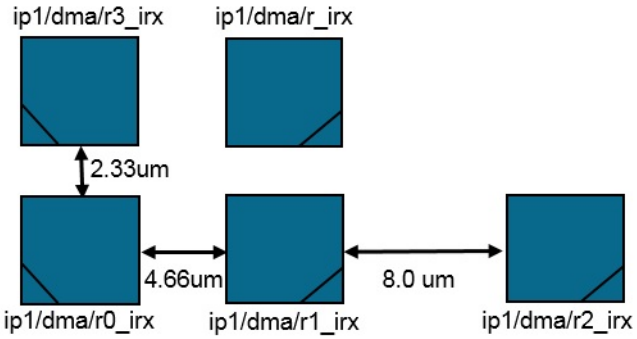
`-orientation string`	Specifies the allowed orientation for the macros. **Note**: Specified orientation overwrites the previous definition. *Data_type*: string, optional

`-parallel_run_length` *float*	Specifies the parallel run length to enable space checking. **Note:** When the `-parallel_run_length` parameter is specified along with the `-horizontal_stacking` or `-vertical_stacking` parameters, only the macros with parallel run length >= the specified value will be counted in the stacking length. *Minimum=0* *Maximum=100000* *Data_type*: float, optional
`-pg_resource_model {`*M1 value1 M2 value2 …*`}`	
	Specifies the pg resource model for concurrent macro placement. During the concurrent macro placement the power mesh does not exist. This parameter allows you to mimic the number of routing tracks occupied by the real power mesh to make the congestion analysis more accurate during the concurrent macro placement. **Note:** This setting is generated automatically by the `create_pg_model_for_macro_place` command. It is highly recommended to use this command to define the `-pg_resource_model` parameter. *Data_type*: string, optional
`-place_order {horizontal | vertical}`	Specifies the place order in row (horizontal) or column (vertical) first for macro array. *Default*: `horizontal` *Data_type*: enum, optional
`-same_length_site` *integer*	Specifies the minimum number of sites between the vertical edge of two macros. *Minimum=0* *Maximum=100000* *Data_type*: int, optional
`-track_adjustment {{`*layer1 density1*`} {`*layer2 ….*`}}…`	Specifies the layer density to add track adjustment on each macro. For each routing layer and each macro, if the PG density over a macro is lower than the standard cell area then it increases the track resource in macro area to mimic the lower PG density. **Note:** This setting is generated automatically by the `create_pg_model_for_macro_place` command. It is highly recommended to use this command to define the `-cell_obs` parameter. *Data_type*: string, optional
`-valid_group_orients {r0 mx my}`	Specifies the valid orientations for the whole group. *Data_type*: (enum)+, optional
`-vertical_stacking {`*max_macro_stack_length min_space_between_macro_stack max_space_between_macros*`}`	
	Specifies the maximum vertical stack depth and the minimum vertical spacing. • *max_macro_stack_length:* Specifies that the macro stacking length should not be more than the specified value. • *min_space_between_macro_stack:* Specifies the spacing between different macro stacks. • *max_space_between_macros:* Specifies the maximum space between macros. When the macro spacing is less than the specified value, the macros are counted in the stacking length. For example, if the max_space_between_macro value is 9 then the macros will be considered as packed if the space between the macros is between 0 and 9, however, if the real space is 10 then nothing will happened. *Data_type*: string, optional

Example

The following command creates a macro array, ram_array, and defines it constraints.

```
set_macro_place_constraint -array ram_array {{ip1/dma/r0_irx:R0 4.66 ip1/dma/r1_irx:MY 8.0 ip1/dma/r2_irx:MY} 2.33 {ip1/dma/r3_irx:R0 4.66 ip1/dma/r_irx:MY}} -valid_group_orients {r0 my}
```



Related Information

- `check_macro_place_constraint`
- `create_pg_model_for_macro_place`
- `get_macro_place_constraint`
- `reset_macro_place_constraint`
- `write_macro_place_constraint`
- Floorplanning the Design chapter in the *Innovus Stylus Common UI User Guide*
- Using the Mixed Placer chapter in the *Innovus Stylus Common UI User Guide*

set_module_size_fixed

```
set_module_size_fixed
[-help]
-modules
```

Specifies that during floorplan resizing, the size of the specified modules and/or blackboxes should not change. Use this command before specifying the floorplan.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `set_module_size_fixed` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man set_module_size_fixed`
`-modules`	Specifies that the size of the *specified* modules and/or blackboxes should not change during floorplan resizing. Provide a space-separated list of instances. *: Specifies that the size of *all* the modules and/or blackboxes should not change during floorplan resizing. *Data_type*: list, required

Example

The following command specifies that the size of `instanceA` and `instanceB` should not change during floorplan resizing.

```
set_module_size_fixed instanceA instanceB
```

Related Information

- `reset_fixed_module_size`
- Floorplanning the Design chapter in the *Innovus Stylus Common UI User Guide*

set_resize_line

```
set_resize_line
[-help]
-clear_all
[-direction {H|V}]
[{x y}]
[-width float]
```

Sets the resize lines for the `resize_floorplan_shift_based` attribute. The resize lines can be non-continuous though they must be orthogonal. If resize lines are specified, the floorplan will be resized between the area specified by the resize lines (for shift-based floorplan resize). To shrink or expand the floorplan in the horizontal direction, specify a vertical resize line. To shrink or expand the floorplan in the vertical direction, specify a horizontal resize line.

Each resize line can have multiple segments (for example, horizontal, vertical, and again horizontal) but it can only be applied to one direction. If you want to shrink or expand the floorplan in both directions, specify resize lines for each direction. You can create multiple resize lines. The resize lines are removed once the floorplan is resized. You can use this command after importing the design.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `set_resize_line` parameter. For a detailed description of the command and all of its parameters, use the man command: `man set_resize_line.`
`-clear_all`	Clears all resize lines. *Data_type*: bool, optional
`-direction {H|V}`	Specifies whether the resize line is horizontal (H) or vertical (V). *Data_type*: string, optional
`{x y}`	Specifies the resize line point list. *Data_type*: pt list, optional
`-width float`	Specifies the width of the resize line in microns. A positive value expands the floorplan and a negative value shrinks the floorplan. *Data_type*: float, optional

Examples

- The following command sets a resize line with a width of 20 microns in the horizontal direction. As the width has a positive value, the floorplan will expand. The resize line is horizontal and will, therefore, expand the floorplan in a vertical direction.
  ```
  set_resize_line -direction H -width 20 (1924 3476) (3807 3476)
  ```
- The following command sets a resize line with a width of -20 microns in the horizontal direction. As the width has a negative value, the floorplan will shrink. The resize line is horizontal and will, therefore, shrink the floorplan in a vertical direction.
  ```
  set_resize_line -direction H -width -20 (1924 3476) (3807 3476)
  ```
- The following command sets a resize line with a width of 20 microns in the vertical direction. As the width has a positive value, the floorplan will expand. The resize line is vertical and will, therefore, expand the floorplan in a horizontal direction.
  ```
  set_resize_line -direction V -width 20 (2761 2215) (2761 3810)
  ```
- The following command sets a resize line with a width of -20 microns in the vertical direction. As the width has a negative value, the floorplan will shrink. The resize line is vertical and will, therefore, shrink the floorplan in a horizontal direction.
  ```
  set_resize_line -direction V -width -20 (2761 2215) (2761 3810)
  ```
- The following command sets a resize line with a width of -20 microns in the vertical direction. As a * has been specified for one of the coordinates, the value of that coordinate will be automatically derived.
  ```
  set_resize_line -direction V -width -20 (2761 *) (2761 3810)
  ```
- The following command creates a resize line with multiple segments.
  ```
  set_resize_line -direction V -width -20 (2643 1744) (2643 3411) (3382 3411)
  (3382 4117)
  ```

Related Information

- `create_resize_blockage`
- `resize_floorplan`
- `resize_floorplan` Category Attributes
- Floorplan Menu chapter in the *Innovus Stylus Common UI Menu Reference*
- Floorplanning the Design chapter in the *Innovus Stylus Common UI User Guide*
  - Resizing the Floorplan

set_route_blockage_default_layer

```
set_route_blockage_default_layer
[-help]
{{[-layers layer_name | {layer_name_list} | all]
[-cut_layers layer_name | {layer_name_list} | all]
[-trim_metal_layers layer_name | {layer_name_list} | all]
[-drc_region_layers layer_name | {layer_name_list} | all]} | {-all_metal_cut }}
```

Specifies the default layers for routing blockages, for both metal layers and cut layers. The default layers set with this command are applicable for all routing blockages subsequently created using the GUI or the `create_route_blockage` command. Use this command before creating routing blockages.

Note: If any layers are specified with the `create_route_blockage` command, the layers specified with the `set_route_blockage_default_layer` command are ignored.

Parameters

`-all_metal_cut`	
	Specifies that default routing blockages will be created for all metal layers and all cut layers. *Data_type*: bool, required
`-cut_layers layer_name | {layer_name_list} | all`	
	• *layer_name:* Specifies that default routing blockages will be created for the specified cut layers. • `all:` Specifies that default routing blockages will be created for all cut layers. *Data_type*: list, optional
`-drc_region_layers layer_name | {layer_name_list} | all`	
	• *layer_name:* Specifies that default routing blockages will be created for the specified drc region layers. • `all:` Specifies that default routing blockages will be created for all drc region layers. *Data_type*: list, optional
`-help`	Outputs a brief description that includes the type and default information for each `set_route_blockage_default_layer` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man set_route_blockage_default_layer`
`-layers layer_name | {layer_name_list} | all`	
	• *layer_name*: Specifies that default routing blockages will be created for the specified metal layers. • `all:` Specifies that default routing blockages will be created for all metal layers. *Data_type*: list, optional
`-trim_metal_layers layer_name | {layer_name_list} | all`	
	• *layer_name*: Specifies that default routing blockages will be created for the specified trim metal layers. • `all:` Specifies that default routing blockages will be created for all trim metal layers. *Data_type*: list, optional

Example

The following command sets default routing blockages for the metal layers1, 2, and 3 and the cut layers 3 and 5.

```
set_route_blockage_default_layer -layers 1 2 3 -cut_layers 3 5
```

Related Information

- `create_route_blockage`
- `delete_route_blockages`
- Floorplanning the Design chapter in the *Innovus Stylus Common UI User Guide*

snap_floorplan

```
snap_floorplan
[-help]
{{-all} | {-selected} |
{[-guide][-block][-std_cell][-io_pad][-area_io][-partition_core][-pin_guide][-route_blockage][-place_blockage][-macro_pin][-pin][-
pin_blockage]}}
```

Snaps floorplan objects to the grid. You can use this command after importing the design.

Note: Innovus honors the FinFET grid and automatically checks for the FinFET definition. If it detects a FinFET definition, it enables support and sets the placement grid as FinFET grid by default. Since the height of all placeable objects must be a multiple of FinFET grid and the origin of these objects must be snapped to the FinFET grid. The snap_floorplan command is used to snap blocks, I/O pads, standard cells to the FinFET grid. It honors user specified snapping rule for x direction. For y direction, it honors the FinFET grid (when the FinFET grid is horizontal in regular horizontal row design).

Parameters

`-all`	Specifies the snapping of all supported object types. *Data_type*: bool, required
`-area_io`	Specifies the snapping of I/O cells. *Data_type*: bool, optional
`-block`	Specifies the snapping of floorplan blocks. *Data_type*: bool, optional
`-guide`	Specifies the snapping of floorplan guides, regions, and fences. *Data_type*: bool, optional
`-help`	Outputs a brief description that includes the type and default information for each `snap_floorplan` parameter. For a detailed description of the command and all of its parameters, use the man command: `man snap_floorplan`
`-io_pad`	Specifies the snapping of I/O pads. *Data_type*: bool, optional
`-macro_pin`	Specifies the snapping of macro pins to the correct color track. **Note:** The `snap_floorplan` command only snaps signal pins and places a macro with macro pins on color track. Innovus snaps the port on the lowest metal layer which is close to macro origin to the color track. *Data_type*: bool, optional
`-pin`	Specifies the snapping of pins. *Data_type*: bool, optional
`-pin_blockage`	Specifies the snapping of pin blockages. *Data_type*: bool, optional
`-pin_guide`	Specifies the snapping of pin guides. *Data_type*: bool, optional
`-place_blockage`	Specifies the snapping of placement blockages. *Data_type*: bool, optional
`-partition_core`	Specifies the snapping of the partition core. *Data_type*: bool, optional
`-route_blockage`	Specifies the snapping of routing blockages. *Data_type*: bool, optional
`-selected`	Specifies the snapping of all supported object types that are selected in the design display area. *Data_type*: bool, required
`-std_cell`	Specifies the snapping of standard cells. *Data_type*: bool, optional

Related Information

- `snap_floorplan_io`
- Floorplanning the Design chapter in the *Innovus Stylus Common UI User Guide*

snap_floorplan_io

```
snap_floorplan_io
[-help]
[-selected]
[-user_grid | -to_io_row [-overlap_row_only]]
```

Snaps I/O cells to a user-defined grid. Use this command after importing the design.

Note: You can specify the user-defined grid in the *Preferences - Floorplan* form in the View Menu of the Innovus GUI.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `snap_floorplan_io` parameter. For a detailed description of the command and all of its parameters, use the man command: `man snap_floorplan_io`
`-overlap_row_only`	Snaps the I/O cell only onto an overlapping I/O row. If an I/O row is not found, the command is not executed. *Data_type*: bool, optional
`-selected`	Applies to the selected I/Os only. *Default*: snaps to all I/O cells. *Data_type*: bool, optional
`-to_io_row`	Snaps the I/O cell onto an overlapping I/O row first. If an I/O row is not found, places the I/O cell in any row having the same site. *Data_type*: bool, optional
`-user_grid`	Specifies the snapping of I/O pads to user-defined grids. *Default*: snaps I/O to manufacture grid. *Data_type*: bool, optional

Related Information

- `snap_floorplan`
- Floorplanning the Design chapter in the *Innovus Stylus Common UI User Guide*

space_selected

```
  space_selected
[-help]
-fix_side {left | right | center | top | bottom | middle | dist_horizontal | dist_vertical}
[-honor_halo] [-space value [-is_space_lower_left_to_lower_left ] ]
```

Spaces objects (instances, modules, or blockages) horizontally or vertically by a specified distance value, and can evenly distribute the spacing horizontally or vertically between three or more objects. You can use this command after selecting two or more objects in the design display area.

Parameters

`-fix_side`	Specifies the vertical and horizontal object spacing. Choose one of the following: • `left`: Specifies that the left most object is fixed and all other objects are shifted horizontally to the right side of the fixed object, maintaining space defined in the `-space` option. • `right`: Specifies that the right most object is fixed and all other objects are shifted horizontally to the left side of the fixed object, maintaining space defined in the `-space` option. • `center`: Maintains the center point of the selected objects. It calculates the mid-point between the leftmost edge and rightmost edge of the selected objects and then spaces the blocks at the specified distance and centers them around this mid-point so that the left most edge and rightmost edge are at an equal distance from the mid-point. • `top`: Specifies that the top most object is fixed and all other objects are shifted vertically to the down side of the fixed object, maintaining space defined in the `-space` option. • `bottom`: Specifies that the bottom most object is fixed and all other objects are shifted vertically to the up side of the fixed object, maintaining space defined in the `-space` option. • `middle`: Maintains the center point of the selected objects. It calculates the mid-point between the top most edge and bottom most edge of the selected objects and then spaces the blocks at the specified distance and centers them around this mid-point so that the top most edge and bottom most edge are at an equal distance from the midpoint. • `dist_horizontal`: Specifies an even horizontal distribution. • `dist_vertical`: Specifies an even vertical distribution. *Data_type*: enum, required
`-help`	Outputs a brief description that includes the type and default information for each `space_selected` parameter. For a detailed description of the command and all of its parameters, use the man command: `man space_selected`
`-honor_halo`	Honors placement halo. With this option on, selected objects are spaced by a given value measured from the edge of the halo and not from the edge of the object. *Data_type*: bool, optional
`-is_space_lower_left_to_lower_left`	
	When specified it signifies that the spacing value denotes the distance from the Object A's lower left to Object B's lower left. By default, this parameter is false. This implies that the spacing value signifies the distance between the bounding of Object A and Object B. *Data_type*: bool, optional
`-space value`	Specifies the spacing value, in micrometers. *Data_type*: float, optional

Example

The following command fixes the bottom-most highlighted object and moves all the other objects vertically to the top, spacing the objects 5 micrometers apart from each other:

```
space_selected -fix_side bottom -space 5
```

Related Information

- `align_selected`
- `flip_or_rotate_obj`
- Floorplanning the Design chapter in the *Innovus Stylus Common UI User Guide*

split_rectilinear_inst

```
split_rectilinear_inst
[-help]
-inst inst_name
{{-ref_inst inst_name [-spacing_x value] [-spacing_y value]} |
{-cut_box {x1 y1 x2 y2}} | {-corner corner_name [-x value -y value]}}
[-push_out side_edge_name]
```

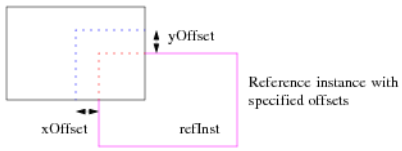
Cuts partition or black box boundaries to rectilinear instances that fit blocks in a given die, without modifying the existing block area. The command performs the rectilinear cut based on the following criteria that you specify:

- The name of a hierarchical instance.
- The name of a referenced instance.
- The name of corner where the cut is to be defined.
- The x and y offset values, in microns, from a specific corner or an overlapping area.
- The name of a side or edge that is to be pushed out, to maintain the existing block area.

You can use the command at any point in the design flow

Parameters

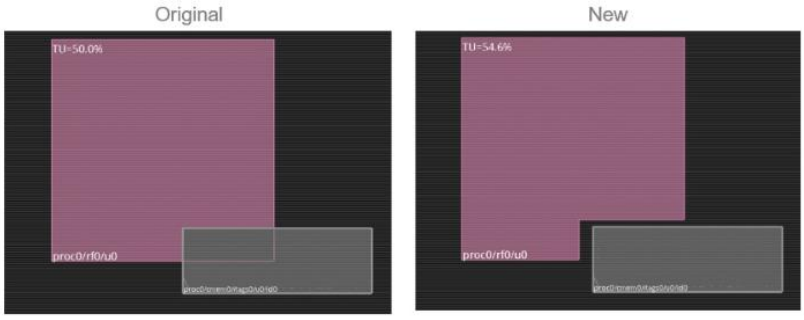
`-corner corner_name`	Specifies the name of a corner from where the cut is to be defined. This can be lx, ly, ux, uy coordinates, or edge number in case of a rectilinear corner. Edge numbers are counted clock-wise from the lower left corner starting with 0. *Data_type*: string, optional
`-cut_box {x1 y1 x2 y2}`	Specifies the rectilinear box that is to be cut. • `x1`: Specifies the lower left x coordinate of the cut area. • `y1`: Specifies the lower left y coordinate of the cut area. • `x2`: Specifies the upper right x coordinate of the cut area. • `y2`: Specifies the upper right y coordinate of the cut area. *Data_type*: rect, optional
`-help`	Outputs a brief description that includes the type and default information for each `split_rectilinear_inst` parameter. For a detailed description of the command and all of its parameters, use the man command: `man split_rectilinear_inst`
`-inst inst_name`	Specifies the name of a hierarchical instance. The instance can be a module, an uncommitted partition or a black box. *Data_type*: (inst|hinst|group)+, required
`-push_out side_edge_name`	Specifies the x offset value, in microns, from a specified corner. Corner numbers are counted clock-wise from the lower left corner starting with 0. This can be: • top • bottom • left • right • rectilinear edge number **Note**: If you do not specify a side or edge, the `split_rectilinear_inst` command will not maintain the existing area and all the sides or edges are modified by default. *Data_type*: string, optional
`-ref_inst inst_name`	Specifies the name of a reference instance. The instance can be a module, an uncommitted partition, a black box or a hard macro. *Data_type*: (inst|hinst|group)+, required

`-spacing_x value`	Specifies the x offset value, in microns, of the overlapping area between the specified block and the reference object. *Default*: 0  *Data_type*: float, optional
`-spacing_y value`	Specifies the y offset value, in microns, of the overlapping area between the specified block and the reference object. *Default*: 0 *Data_type*: float, optional
`-x value`	Specifies the x offset value, in microns, from a specified corner. Corner numbers are counted clock-wise from the lower left corner starting with 0. *Data_type*: float, optional
`-y value`	Specifies the y offset value, in microns, from a specified corner. *Data_type*: float, optional

Example

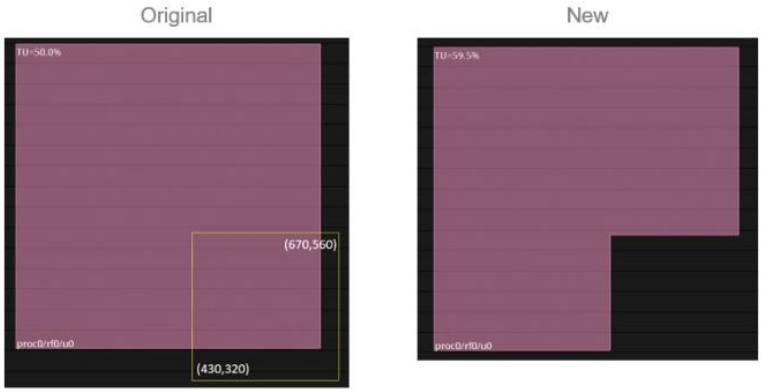
- The following command cuts away the area of proc0/rf0/u0 which overlaps proc0/cmem0/itags0/u0/id0, plus an additional 30um in the X-direction and 15um in the Y-direction resulting in the shape below:

```
split_rectilinear_inst -inst proc0/rf0/u0 -ref_inst proc0/cmem0/itags0/u0/id0 -spacing_x 30 -spacing_y 15
```



- The following command cuts away the intersection of the module and cut box resulting in the shape below:

```
split_rectilinear_inst -inst proc0/rf0/u0 -cut_box 430 320 670 560
```

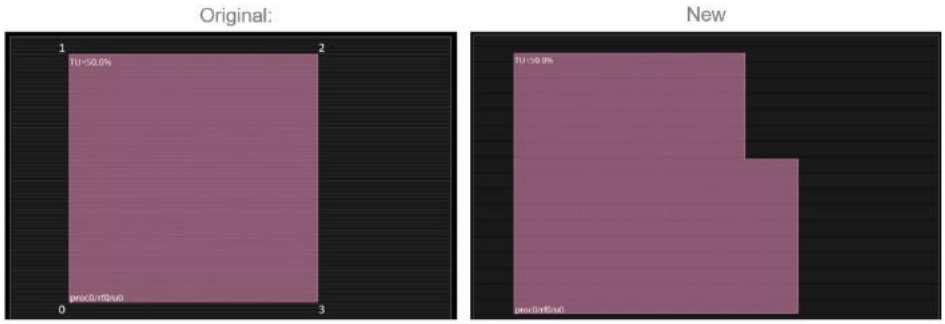


- The following command ccuts the module at corner number 2. The cut is 100um in the X-direction and 200um in the Y-direction:

```
split_rectilinear_inst -inst proc0/rf0/u0 -corner 2 -x 100 -y 200
```

- The command following cuts the module at corner number 2. The cut is 100um in the X-direction and 200um in the Y-direction. The area is maintained by pushing out the left edge (notice utilization of 50% is maintained):

```
split_rectilinear_inst -inst proc0/rf0/u0 -corner 2 -x 100 -y 200 -push_out left
```



Related Information

- Floorplanning the Design chapter in the *Innovus Stylus Common UI User Guide*

split_row

```
split_row
[-help]
[-area box | -row_id string | -selected | -from_io {left bottom right top}]
[-halo float| {-left_gap float| -right_gap float | -top_gap float | -bottom_gap float}]
[-site name]
[-keep_cell]
```

Cuts site rows within an area or that intersect with the selected object(s). Use this command after rows have been created. Only insts, hinsts, place blockages, route blockages, power domains, groups, or bumps can participate in row splitting.

Note: The default behavior is to cut rows that intersect with placement blockages and macros.

Parameters

`-area box`	Rows will be cut within the specified box. `box` is a tcl list containing 4 items. The items are, in order: `LLX, LLY, URX, URY.` **Note:** The `-halo` parameter cannot be applied `-area` used. *Data_type*: rect, optional
`-bottom_gap float`	Specifies the additional gap to be provided on the bottom side of the specified or selected object. *Data_type*: float, optional
`-from_io {left bottom right top}`	
	Cuts core rows the specified distance from I/O cells. `left` is the keep-away distance from IO cells on left side of die box, and so forth. *Data_type*: rect, optional
`-halo float`	Specifies additional space to be provided on the top, bottom, left, and right sides of the objects causing rows to be cut. Rows are cut to the expanded boundary. The same value is used for all sides. *Data_type*: float, optional
`-help`	Outputs a brief description that includes the type and default information for each `split_row` parameter. For a detailed description of the command and all of its parameters, use the man command: `man split_row`
`-keep_cell`	Specifies that all cells that are placed inside the cut row area will not be unplaced. *Default*: All cells inside the cut rows will be unplaced. *Data_type*: bool, optional
`-left_gap float`	Specifies the additional gap to be provided on the left side of the specified or selected object. *Data_type*: float, optional
`-right_gap float`	Specifies the additional gap to be provided on the right side of the specified or selected object. *Data_type*: float, optional
`-row_id string`	Specifies the name of an intersecting object. Rows that intersect this object are cut. Only one name may be specified. *Data_type*: bool, optional
`-selected`	Specifies that the rows that intersect with the selected object(s) will be cut. **Note:** Only supports inst, hinst, power domain, group, place blockage, route_blockage, and bump as selected object(s). *Data_type*: bool, optional
`-site name`	Rows containing site `name` will be cut. *Default*: All rows in the core box are cut. *Data_type*: string, optional
`-top_gap float`	Specifies the additional gap to be provided on the top side of the specified or selected object. *Data_type*: float, optional

Example

- The following command splits the rows containing sites of type `SITE_FE283001` in the specified area:
  ```
  split_row -site SITE_FE283001 -area 124.975 138.123 237.378 223.38
  ```
- The following command splits the rows intersecting with selected objects, with a 0.5 micron halo:
  ```
  split_row -selected -halo 0.5
  ```
- The following command splits the rows intersecting with the object named `fuu`, with additional bottom gap of 0.5 microns:
  ```
  split_row -row_id fuu -bottom_gap 0.5
  ```

Related Information

- `create_row`
- `delete_row`
- `init_core_rows`
- `stretch_rows`
- Floorplanning the Design chapter in the *Innovus Stylus Common UI User Guide*

stretch_rows

```
stretch_rows
[-help]
{-left | -right | -left -right}
[-to_core_edge]
```

Stretches selected rows. You can use this command after importing a design.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `stretch_rows` parameter. For a detailed description of the command and all of its parameters, use the man command: `man stretch_rows`
`-left`	Specifies that the left edge of all selected rows should be aligned to the left-most edge among all selected rows. *Data_type*: bool, optional
`-right`	Specifies that the right edge of all selected rows should be aligned to the right-most edge among all selected rows. *Data_type*: bool, optional
`-to_core_edge`	Specifies that row edges should be aligned to the core boundary. *Data_type*: bool, optional

Example

The following command specifies that the left edge of all selected rows should be aligned to the left-most edge among all selected rows.

```
stretch_rows -left
```

Related Information

- `create_row`
- `delete_row`
- `init_core_rows`
- `split_row`
- Floorplanning the Design chapter in the *Innovus Stylus Common UI User Guide*
  - Creating and Editing Rows

swap_pins

```
swap_pins
[-help]
```

Switches the locations of two pins. You can use this command after selecting two pins in the same module in the design display area.

Parameters

`-help`	Outputs a brief description of the `swap_pins` command. For a detailed description of the command, use the man command: `man swap_pins`

Related Information

- Floorplanning the Design chapter in the *Innovus Stylus Common UI User Guide*
  - Editing Pins

trace_obj_connectivity

```
trace_obj_connectivity
[-help]
[-clear_trace]
[-level level]
[-mode {netlist_based timing_based}]
[-out_file file_name]
[-insts string] | [-ports string] | [-selected ] | [-in_files file_name]
```

Traces the connections of specified macros, selected macros, or specified ports. You can also trace the connections based on netlist or timing. It supports different levels. You can also set global parameters for the `trace_obj_connectivity` by using the `trace_obj` Category Attributes. The attributes that you specify are then used automatically whenever you run the `trace_obj_connectivity` command or use the *Trace Macro* form. You can use this command after `proto_design`, `place_design`, or even after the whole design is completed.

Parameters

`-clear_trace`	Clears the existing trace result. *Data_type*: bool, optional
`-help`	Outputs a brief description that includes type and default information for each `trace_obj_connectivity` parameter. For a detailed description of the command and all of its parameters, use the man command: `man trace_obj_connectivity`
`-in_files file_name`	Specifies the name of the file from which saved trace macro result is loaded. *Data_type*: string, optional
`-insts string`	Specifies the macros to be traced. *Data_type*: string, optional
`-level level`	Specifies the trace level. *Maximum Level*: 10 *Minimum Level*: 1 *Default*: 1 *Data_type*: int, optional
`-mode {netlist_based timing_based}`	
	Specifies the trace mode to use: • `netlist_based`: Specifies that connection of the macro will be traced based on the netlist. • `timing_based`: Specifies that connection of the macro will be traced based on advanced timing tcl scripts command `all_fanin` and `all_fanout`. *Data_type*: enum, optional
`-out_file file_name`	Specifies the name of the file in which the trace macro results will be saved. *Default*: `<top_module_name>.tmcdb` *Data_type*: string, optional
`-ports string`	Specifies the names of the ports to be traced. *Data_type*: string, optional
`-selected`	Traces the selected macros or ports. *Data_type*: bool, optional

Example

The following command traces the connections for blk1 upto level 3:

```
trace_obj_connectivity -insts blk1 -level 3
```

Related Information

- `trace_obj` Category Attributes
- `gui_show_obj_connectivity`
- `all_fanin`
- `all_fanout`.
- Floorplanning the Design chapter in the *Innovus Stylus Common UI User Guide*
- Floorplan Menu chapter in the *Innovus Stylus Common UI Menu Reference*

trace_obj Category Attributes

The root attributes in the `trace_obj` category are:

- trace_obj_macro_pins
- trace_obj_max_fanin_fanout
- trace_obj_register_inputs
- trace_obj_register_outputs

To set the value of any root attribute, use the `set_db` command:

```
set_db attribute_name value
```

To get the current value of any root attribute, use the `get_db` command:

```
get_db attribute_name
```

To get the current value of all attributes in the `trace_obj` category, use the following command:

```
get_db -category trace_obj
```

trace_obj_macro_pins

`trace_obj_macro_pins` *list_of_pins*

Default: " "

Data_type: string, read/write

Specifies the macro pin name patterns.

Related Commands

- `trace_obj_connectivity`

trace_obj_max_fanin_fanout

`trace_obj_max_fanin_fanout` *number*

Default: 1000

Data_type: int, read/write

Specifies the maximum fanin/fanout number for a pin. While tracing the connectivity, a pin will be ignored if its fanin/fanout number is larger than the specified number.

Related Commands

- `trace_obj_connectivity`

trace_obj_register_inputs

`trace_obj_register_inputs` *list_of_pins*

Default: " "

Data_type: string, read/write

Specifies the register input pin names.

Related Commands

- `trace_obj_connectivity`

trace_obj_register_outputs

`trace_obj_register_outputs` *list_of_pins*

Default: " "

Data_type: string, read/write

Specifies the register output pin names.

Related Commands

- `trace_obj_connectivity`

unplace_obj

```
unplace_obj
[-help]
{-all  | -insts  | -blocks  | -groups  | -hinsts  | objNameList}
```

Unplaces objects. It clears the specified constraints (blocks, groups, guides, regions, and fences) from the floorplan. You can use this command after importing the design.

Parameters

`-all`	Unplaces all objects from the floorplan, including instances, blocks and groups. *Data_type*: bool, optional
`-blocks`	Unplaces all blocks from the floorplan. *Data_type*: bool, optional
`-groups`	Unplaces all the instance groups from the floorplan. This parameter will disable constraints, such as guides, fences, and regions. *Data_type*: bool, optional
`-help`	Outputs a brief description of the `unplace_obj` command. For a detailed description of the command, use the man command: `man unplace_obj`
`-hinsts`	Unplaces all the hierarchical instances from the floorplan. This parameter will disable constraints, such as guides, fences, and regions. *Data_type*: bool, optional
`-insts`	Unplaces all the standard cell instances from the floorplan. *Data_type*: bool, optional
objNameList	Specifies the object list. This can be a mixed list of instances, blocks, or groups, or their names. *Data_type*: string, optional

Examples

- The following command will unplace all the blocks.
  ```
  unplace_obj -blocks
  ```
- The following command will unplace the groups `group1` and `group2`.
  ```
  unplace_obj {group1 group2}
  ```

Related Information

- `copy_obj`
- `paste_obj`
- `move_obj`
- Floorplanning the Design chapter in the *Innovus Stylus Common UI User Guide*

update_floorplan

```
update_floorplan
[-help]
[-no_snap_to_grid]
[[-core_to_left value][-core_to_bottom value][-core_to_right value][-core_to_top value] |
[-core_to_edge {left bottom right top}]]
```

Changes the distances between the core box and the die box on the sides you specified. Use this command after specifying the floorplan data or running an initial floorplan.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `update_floorplan` parameter. For a detailed description of the command and all of its parameters, use the man command: `man update_floorplan`
`-core_to_bottom value`	Specifies the distance from the bottom of the core box to the bottom of the die box. *Data_type*: float, optional
`-core_to_edge {left bottom right top}`	
	Specifies the distances between the core box and die box on each side (left, bottom, right, and top). *Data_type*: rect, optional
`-core_to_left value`	Specifies the distance from the left of the core box to the left of the die box. *Data_type*: float, optional
`-core_to_right value`	Specifies the distance from the right of the core box to the right of the die box. *Data_type*: float, optional
`-core_to_top value`	Specifies the distance from the top of the core box to the top of the die box. *Data_type*: float, optional
`-no_snap_to_grid`	Specifies that the core box (or die box) boundary will not be snapped to the nearest metal pitch even if the specified die/ core box size is not an integer multiple of the smallest metal pitch. This prevents the input floorplan boxes from being changed. *Data_type*: bool, optional

Example

The following command changes the distance from the bottom of the core box to the bottom of the die box to 200:

```
update_floorplan -core_to_bottom 200
```

Related Information

- Floorplanning the Design chapter in the *Innovus Stylus Common UI User Guide*

update_floorplan_for_macro_place

```
update_floorplan_for_macro_place
[-help]
[-init_rows]
[-keep_fixed_insts {inst1 inst2 ...}]
[-remove {all route_blockage place_blockage relative_floorplan power_domain insts_place wire boundary_constraint physical_inst drc PG_pin}]
```

Enables you to clean the floorplan before defining constraints for macros with the `set_macro_place_constraint` command.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `update_floorplan_for_macro_place` parameter. For a detailed description of the command and all of its parameters, use the man command: `man update_floorplan_for_macro_place`
`-init_rows`	Creates core rows. *Data_type*: bool, optional
`-keep_fixed_insts {inst1 inst2 ...}`	
	Keeps the specified instances in the floorplan. It sets the instances which are not unplaced. *Data_type*: string, optional
`-remove {all route_blockage place_blockage relative_floorplan power_domain insts_place wire boundary_constraint physical_inst drc PG_pin}`	
	Removes the specified objects from the floorplan. *Data_type*: (enum)+, optional

Example

- The following command initializes the core rows and deletes the placement blockages from the floorplan:

  ```
  update_floorplan_for_macro_place -init_rows -remove place_blockage
  ```

Related Information

- `create_pg_model_for_macro_place`
- `get_macro_place_constraint`
- `reset_macro_place_constraint`
- `set_macro_place_constraint`
- `write_macro_place_constraint`
- Floorplanning the Design chapter in the *Innovus Stylus Common UI User Guide*
- Using the Mixed Placer chapter in the *Innovus Stylus Common UI User Guide*

update_floorplan_obj

```
update_floorplan_obj
[-help]
[-obj { hinst | inst| group | place_blockage | route_blockage | resize_blockage | bump | row | port | design}]
{-rects {{x1 y1} {x2 y2}...} | -polygon {{x1 y1} {x2 y2}...} | -move {x y}}
```

Updates the specified floorplan object by moving or reshaping. You can use this command after importing the design.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `update_floorplan_obj` parameter. For a detailed description of the command and all of its parameters, use the man command: `man update_floorplan_obj`
`-move{x y}`	Moves the lower left corner of object to the specified point {x y}. *Data_type*: point, optional
`-polygon {{x1 y1} {x2 y2}...}`	Specifies the corner coordinates `{{x1 y1} {x2 y2}...}`, of the rectilinear object. The rectilinear object can take any shape depending on the number of x and y coordinates you specify. **Note:** The corner coordinates of a rectilinear object should be specified and this option only supports specifying one polygon shape. **Note:** For specifying a set of regions (disjoined) described in polygons, you must first use the `get_computed_shapes` command to get the equivalent in rects and then use the `-rects` parameter to specify the coordinates. *Data_type*: (point)+, optional
`-rects {{x1 y1} {x2 y2}...}`	Specifies one or more rects that must overlap to create one shape. The low left corner coordinate and the upper right corner coordinate of each rect should be specified. For example, `-rects {llx1 lly1 urx1 ury1 llx2 lly2 urx2 ury2 …}` **Note:** You can specify the coordinates of disjoint rects as well. *Data_type*: (rect)+, optional
`-obj { hinst | inst| group | place_blockage | route_blockage | resize_blockage | bump | row | port | design}`	
	Specifies a single pointer to the object that needs to be reshaped/moved. It must be one of the following types: • `hinst`, `group`, `place_blockage`, and `route_blockage` allow both reshape and move. • `inst`, `bump`, `row` and `port` only allow move. • `design` only allows reshape. **Note**: To get the db pointer for an object, you can select the object in GUI then use the `get_db selected` command. For example, `get_db selected` returns: `hinst:DTMF_CHIP/DTMF_INST/RAM_128x16_TEST_INST` *Data_type*: (hinst|inst|group|place_blockage|route_blockage|resize_blockage|bump|row|port|design), required

Example

- The following command gets the db pointer for the object to be updated and then updates it to the specified points:
  ```
  get_db selected
  ```
  returns: `hinst:DTMF_CHIP/DTMF_INST/RAM_128x16_TEST_INST`
  ```
  update_floorplan_obj -obj hinst:DTMF_CHIP/DTMF_INST/RAM_128x16_TEST_INST -rects {500 500 700 700}
  ```

Related Information

- `get_db`
- Floorplanning the Design chapter in the *Innovus Stylus Common UI User Guide*

update_group

```
update_group
[-help]
{-name group_name {-add | -delete } -objs object_name}
```

Adds or deletes instances, hierarchical instances, or groups to/from a specified group. A group is a user-defined pseudo hierarchy and is created to contain instances, hierarchical instances and/or other groups. Use this command after creating a group using the `create_group` command.

Note:

- When you add a member to a group whose boundary constraint is a fence or a region, the size of the bounding box of the group does not change.
- If the group being added is also a power domain, a warning is displayed and the command is not executed.

Parameters

`-add`	When specified, it adds objects to the group. *Data_type*: bool, optional
`-delete`	When specified, it deletes objects to the group. *Data_type*: bool, optional
`-help`	Outputs a brief description that includes the type and default information for each `update_group` parameter. For a detailed description of the command and all of its parameters, use the man command: `man update_group`
`-name group_name`	Specifies the name of the group to modify. *Data_type*: (group), required
`-objs object_name`	Specifies the name of the object, another group, hierarchical instance or leaf instance. **Note:** You can specify an instance names list in the { *inst_name_list* } format. The names list option is used for instances only and NOT for group or hierarchical instances. *Data_type*: (inst|hinst|group)+, required

Examples

- The following command adds hierarchical instance to group adder1:
  `update_group -name adder1 -add -objs SH28/I12/I44`
- The following command adds group subadder to group adder1:
  `update_group -name adder1 -add -objs subadder`
- The following command adds the instances inst_I1, inst_I2, and inst_I3 to group inst_group1:
  `update_group -name inst_group1 -add -objs {inst_I1 inst_I2 inst_I3}`
- The following command deletes the instances inst_I4 and inst_I5 from group inst_group1:
  `update_group -name inst_group1 -delete -objs {inst_I4 inst_I5}`

Related Information

- `create_group`
- Floorplanning the Design chapter in the *Innovus Stylus Common UI User Guide*
  - Grouping Instances

update_obj_floorplan_cut_box

```
update_obj_floorplan_cut_box
[-help]
-name name
-rects box_list
-type type
```

Specifies the rectilinear shape of floorplan objects by describing the cut out area. The floorplan objects include block top cell, module constraint (guide, region, fence), layershape, pin, net, group, and instance. The command creates a rectilinear cut box for an object. A rectilinear object is comprised of two or more boxes. To use this command, divide the required cut out area into one or more bounding boxes and specify the coordinates of each bounding box in the command. Once you run the command, the rectilinear cut area disappears from the object after a redraw. You can use the command at any point in the design flow.

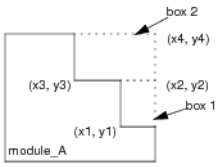
Parameters

`-help`	Outputs a brief description that includes the type and default information for each `update_obj_floorplan_cut_box` parameter. For a detailed description of the command and all of its parameters, use the man command: `man update_obj_floorplan_cut_box`
`-name name`	Specifies the name of the object for type. *Data_type*: string, required
`-rects box_list`	Specifies the rectilinear shape of an object. The box list must be specified in the following format: `{{x1 y1 x2 y2} ....}` *Data_type*: list, required
`-type type`	Specifies the type of object for name. This can be: • `Cell` • `Group` • `Instance` • `Layershape` • `Module` • `Net` • `Pin` *Data_type*: string, required

Example

The following example defines the rectilinear shape of the module module_A by specifying 2 rectilinear cut boxes defined by coordinates (x1, y1), (x2, y2), (x3, y3), and (x4, y4).

```
update_obj_floorplan_cut_box -name module_A -type Module -rects {{x1 y1 x2 y2} {x3 y3 x4 y4}}
```



Related Information

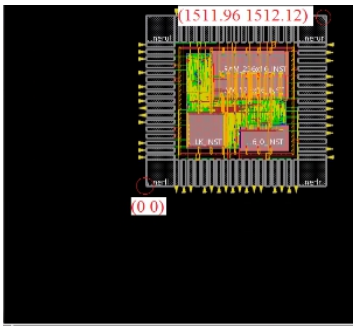
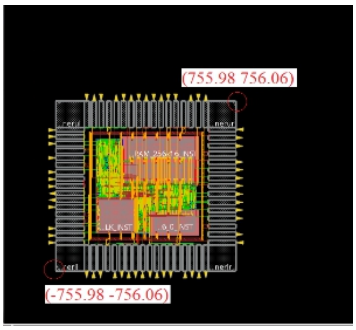
- Floorplanning the Design chapter in the *Innovus Stylus Common UI User Guide*

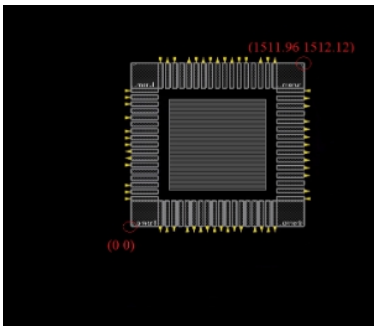
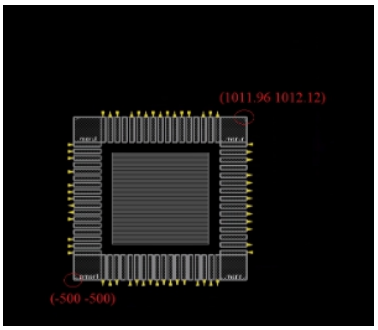
update_origin

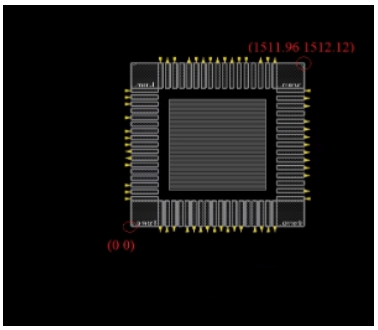
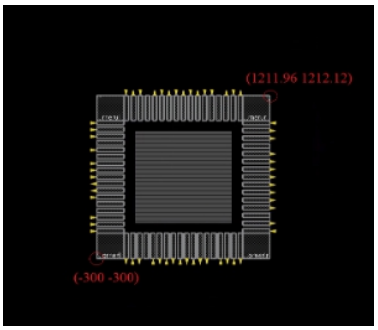
```
update_origin
[-help]
{-center | -lower_left | -location x y | -delta x y}
```

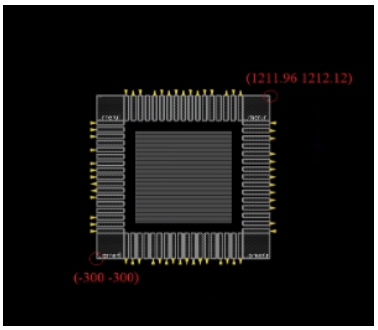
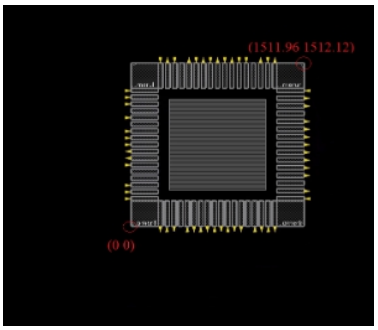
Moves the entire design to a new location. All objects that are recorded in DEF are moved to a new location. No change is made to the other attributes of objects except the location. All properties and characters such as orientation and placement status remain the same after the move.

Parameters

`-center`	Moves the origin point of the entire design to the center point of the current floorplan box. Example: `update_origin -center` Before:  After:  *Data_type*: bool, optional

`-delta x y`	Moves the origin point of the entire design by the specified distance, *deltaX* and *deltaY.* Example: `update_origin -delta 500 500` Before:  After:  *Data_type*: point, optional
`-help`	Outputs a brief description that includes the type and default information for each `update_origin` parameter. For a detailed description of the command and all of its parameters, use the man command: `man update_origin`

`-location` *x y*	Moves the origin point of the entire design to the specified location, *x y*. Example: `update_origin -location 300 300` Before:  After:  *Data_type*: point, optional

`-lower_left`	Moves the origin point of the entire design to the lower left point of the current floorplan box. Example: update_origin `-lowerLeft` Before:  After:  *Data_type*: bool, optional

Example

The following command moves the origin point to the center point of the current floorplan box:

```
update_origin -center
```

Related Information

- Floorplanning the Design chapter in the *Innovus Stylus Common UI User Guide*

update_partition_to_stairway

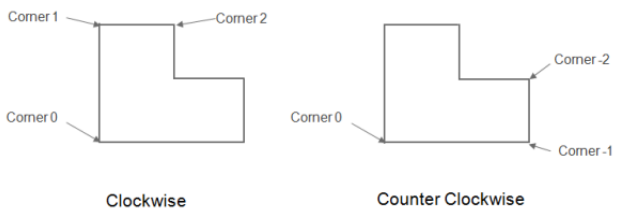
```
update_partition_to_stairway
[-help]
-dx value
-dy value
[-grow_neighbor_partition_direction {x y}]
-partition partition_name
-step number_of_steps
{-corner {ll lr ul ur} | -vertex number_of_vertex}
```

Creates small stairway rectilinear edges at a specified partition corner to improve congestion.

Note: This command does not support master/clones.

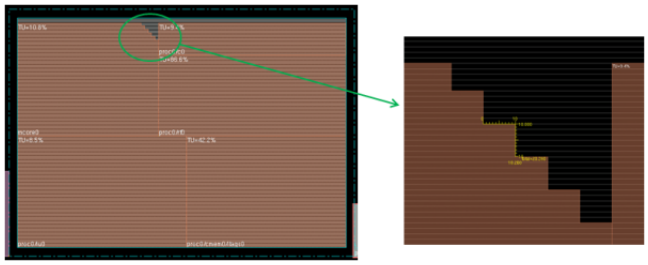
Note: The `-dx`, `-dy`, `-ptn`, `-step`, and `-corner/-vertex` options should be set by default. The partition will snap to the row automatically.

Parameters

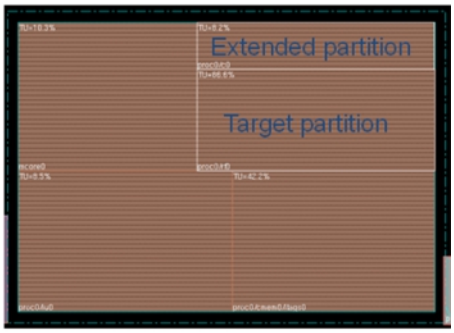
`-help`	Outputs a brief description that includes the type and default information for each `update_partition_to_stairway` parameter. For a detailed description of the command and all of its parameters, use the man command: `man update_partition_to_stairway`
`-corner {ll lr ul ur}`	Specifies partition corner where a stairway will be created. • `ul`: Upper left • `ll`: Lower left • `lr`: Lower right • `ur`: Upper right *Data_type*: enum, optional
`-dx value`	Specifies step width. *Data_type*: float, required
`-dy value`	Specifies step height. *Data_type*: float, required
`-grow_neighbor_partition_direction {x y}`	
	Specifies the direction of the target partition in which the neighbor partition will grow to fill the stairway gap between the target partition and the neighbor partition. If this option is not specified, no abutted partition will grow. *Data_type*: enum, optional
`-partition partition_name`	Specifies the name of the target partition. *Data_type*: string, required
-step *number_of_steps*	Specifies the number of steps to be created in stairway. *Data_type*: int, required
`-vertex number_of_vertex`	Specifies partition vertex where stairway will be created. Vertex starts from the lower-left corner of the partition clockwise. The first vertex is 0. A vertex can be negative, and for a negative vertex, corner numbering starts at 0 from the lower-left corner of a partition counter clockwise.  *Data_type*: int, optional

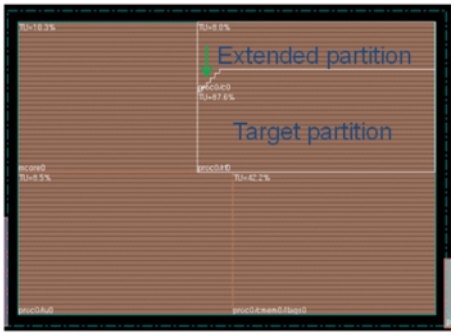
Examples

- The following command creates five-stairway rectilinear edges. The width and height of steps are 10.
  `update_partition_to_stairway -dx 10 -dy 10 -partition mcore -step 5 -corner ur`

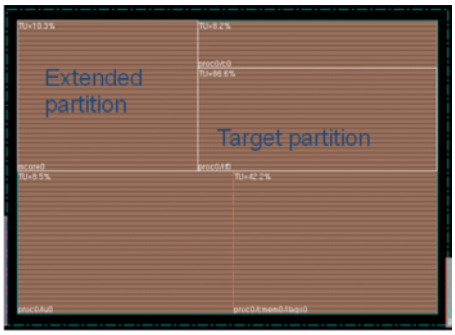


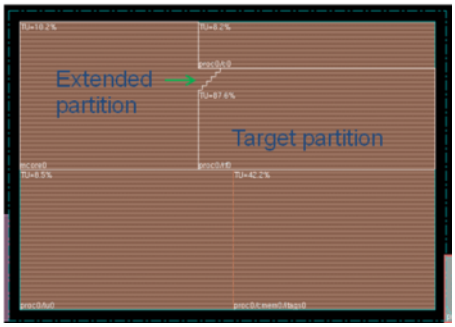
- The following command sets `-grow_neighbor_partition_direction y`, and the partition next to the target partition in the Y direction extends.
  `update_partition_to_stairway -dx 10 -dy 10 -partition regfile_iu_rftype2_abits8_dbits32_words136 -step 5 -corner ul -grow_neighbor_partition_direction y`





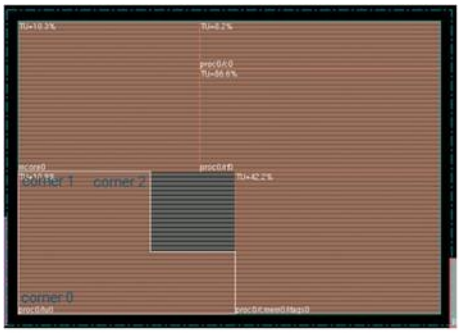
- The following command sets `-grow_neighbor_partition_direction x`, and the partition next to the target partition in the X direction extends.
  `update_partition_to_stairway -dx 10 -dy 10 -partition regfile_iu_rftype2_abits8_dbits32_words136 -step 5 -corner ul -grow_neighbor_partition_direction x`

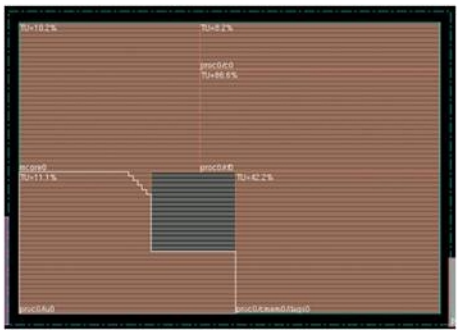




- The following command changes the vertex 2 to a stairway.

```
update_partition_to_stairway -dx 10 -dy 10 -partition iu  -step 5 -vertex 2 -grow_neighbor_partition_direction y
```





Related Information

- Floorplanning the Design chapter in the *Innovus Stylus Common UI User Guide*

update_route_rule

```
update_route_rule
[-help]
[-add_via {via_name1 via_name2 ...}]
[-generate_via]
[-hard_spacing {0|1}]
[-min_cut {layer1[:layer2] min_cut ...}]
-name ruleName
[-spacing {layer1[:layer2] spacing ...}]
[-use_via_cut_class {layer cut_class [nr_row nr_col] ...}]
[-via {via_name1 via_name2 ...}]
[-width {layer1[:layer2] width ...}]
```

Modifies a non-default rule created by the `create_route_rule` command, or coming in from DEF or the OpenAccess design data.

Note: Any NONDEFAULTRULE defined in LEF or the OA techfile cannot be modified by this command. You must modify the LEF or OA techfile directly if you need to change them.

This command should normally be used before any routing exists in the design. If there is any existing routing using this NDR, then the changes may cause new DRC violations. In some cases ecoRoute may be able to repair the routing (for example just changing the vias can often be repaired, and just adding new via choices should not need any repair), but in many cases you will need to reroute the entire design (e.g. when changing the width or spacing) to get good results.

If `add_route_vias_auto` attribute is set to `true`, then vias are automatically generated for the modified NDR. See the add_route_vias Category Attributes chapter for more details.

Parameters

`-help`	Prints a brief description that includes type and default information for each `update_route_rule` parameter. For a detailed description of the command and all of its parameters, use the man command: `man update_route_rule`
`-add_via {`*`via_name1 via_name2 ...`*`}`	
	Specifies a list of vias to add to the existing NDR via list. You can use this parameter to add DFM vias or other custom vias. *Data_type*: string, optional
`-generate_via`	Auto-generates vias with the generateVias command. This is intended for debugging, and not used generally as it only adds vias for the current session. If you save, and later restore the design in a new session, the added vias are not added to the NDR via list any longer. You can use this option to do experiments in the current session, or write out a LEF or DEF file to put the modified NDR into the LEF or OA techfile. **Note**: If the `add_route_vias_auto` attribute is set to `true`, then vias are always auto-generated even when `-generate_via` is not given. This is the normal usage behavior if you want to use the auto-generated vias for the rest of the flow (For example, after save, and then restoring the design in a new session.) *Data_type*: enum, optional
`-hard_spacing {0|1}`	Sets the non-default rule as HARDSPACING. *Data_type*: int, optional
`-min_cut {`*`layer1`*`[:`*`layer2`*`] `*`min_cut ...`*`}`	
	Specifies the minimum cut limit for via layers. *Data_type*: string, optional
`-name `*`ruleName`*	Specifies the name of the non-default rule to be modified. *Data_type*: string, required
`-spacing {`*`layer1`*`[:`*`layer2`*`] `*`spacing ...`*`}`	
	Specifies the spacing for the routing layers. *Data_type*: string, optional
`-use_via_cut_class {`*`layer cut_class [nr_row nr_col] ...`*`}`	
	Resets the layer `cut_class` list. *Data_type*: string, optional
`-via {`*`via_name1 via_name2 ...`*`}`	
	Resets the via list to empty, and then adds the list of named vias to the NDR via list. *Data_type*: string, optional
`-width {`*`layer1`*`[:`*`layer2`*`] `*`width ...`*`}`	
	Specifies the width for the routing layers. *Data_type*: string, optional

Example

- The following command creates a rule that requires double width and double spacing. All other values are the same as the default rule. add_ndr is like update_route_rule for via generation. If the `add_route_vias_auto` attribute is set to `true`, then vias are automatically generated that match the different layer widths. If not, the vias from the default rule are used, which is not correct for wires that are twice the default width.
  add_ndr -name NDR_2w2s -spacing {metal1:metal4 0.2 metal5:metal6 0.4 metal7 0.6} -width {metal1:metal4 0.2 metal5:metal6 0.4 metal7 0.6}

- The following command modifies the width and spacing values in non-default rule NDR_2w2s created by the command. above. If the `add_route_vias_auto` attribute is set to `true`, vias are automatically generated with matching widths and added to the NDR via-list. If not, then the original vias are not changed, which can be wrong for different widths.
  update_route_rule -name NDR_2w2s -spacing {metal1:metal4 0.4 metal5:metal6 0.8 metal7 0.9} -width {metal1:metal4 0.4 metal5:metal6 0.8 metal7 0.9}
- The following command auto-generates vias for rule NDR_2w2s with the generateVias command to get vias that match the wider wire widths, even if `add_route_vias_auto` attribute is set to false. This is only useful for debugging in the current session, or to write a LEF file with the NDR to add back into the LEF or OA techfile. If you want to use the auto-generated vias for the rest of the flow, you should set `add_route_vias_auto` attribute to `true`.
  update_route_rule -name NDR_2w2s -generate_vias
- The following command modifies existing rule NDR_2w2s to specify MINCUTS to 2 for the Vx layers (via1 to via3).
  update_route_rule -name NDR_2w2s -min_cut {via1:via3 2}

Related Information

- `create_route_rule`
- `add_route_vias` Category Attributes
- `add_route_via_defs`
- Floorplanning the Design chapter in the *Innovus Stylus Common UI User Guide*

write_floorplan

```
write_floorplan
file
[-help]
[-no_name]
[-sections {macro | pin | bndry | special_route | pin_constraint}]
[-xml]
```

Saves the floorplan information to a file. The command saves all the floorplan blocks that are placed/fixed/covered, in the floorplan file. When you save a floorplan, the Innovus software saves all objects which can be saved and any specific properties which are affected if the data is saved. By default, the floorplan information is saved as a Version 8 floorplan file. After you save a floorplan, the Innovus software creates the following files:

- A general floorplanning file with the extension .fp
- A power route data file with the extension .fp.spr
  Note: If there is an entry in the IO Cell Libraries field in the Design Import form, a third file is created with the extension `.fp.areaio`.

Note:

- The `write_floorplan` command honors the blackbox min/max aspect ratio information in the floorplan information file.
- The `write_floorplan` command saves net attributes including non-default attributes of a net, like weight, bottomPreferredRoutingLayer, or detour. You can use this command after importing the design.
- The `write_floorplan` command saves the user-specified seed information by default.
- The `write_floorplan` command saves the information related to blockages, tracks, and pin shapes into the *.fp file.
- The `write_floorplan` command saves feedthrou-bumps into the .fp.areaio file. The format is as below:
  `Bump: [BumpName] [BumpCellName] [x coord] [y coord] [BumpPin_1] [PinName_1] [BumpPin_2] [PinName_2]`

Parameters

file	Specifies the name of floorplan file to be saved. *Data_type*: string, required
-help	Outputs a brief description that includes type and default information for each `write_floorplan` parameter. For a detailed description of the command and all of its parameters, use the man command: `man write_floorplan`.
-no_name	Specifies that floorplan object names are not output to the floorplan file when you save the floorplan. *Data_type*: bool, optional
-sections {macro | pin | bndry | special_route | pin_constraint}	
	Specifies the objects section in the floorplan file that will be updated. The sections can be: • `macro` • `pin` • `bndry` • `special_route` • `pin_constraint` **Note:** The `write_floorplan` command supports routing blockages on master slice layers. *Data_type*: string, optional
-xml	Saves the floorplan information in the XML format floorplan file. *Data_type*: bool, optional

Example

The following command writes the current floorplan information to the file `myFloorPlan.fp`:

```
write_floorplan myFloorPlan.fp
```

Related Information

- `create_floorplan`
- `check_floorplan`
- `floorplan` Category Attributes
- `read_floorplan`
- `write_floorplan_script`
- The following section in the Floorplanning the Design chapter of the *Innovus Stylus Common UI User Guide*
  - Saving and Loading Floorplan Data

write_floorplan_script

```
write_floorplan_script
[-help]
out_file
[-append_to_file]
[-compact_row]
[-no_snap_to_grid]
[-selected | -sections {boundary row place_blockages route_blockages pin_blockages groups constraints pins iopad areaio bump partitions
blocks global_net_connect bus_sink_groups black_boxes net_group_and_bus_guide relative_fplan pin_groups pin_guides custom_shape cover_cell}]
```

Allows you to write out the specified floorplan sections and source the output file after `init_design`. With this command the tool would write a file that you can examine, edit, and then source to selectively (or completely) define your floorplan. You can specify the floorplan section to write out. The `write_floorplan_script` command writes out a file with Tcl commands to recreate the floorplan. It can optionally write out specific floorplan sections listed with option -sections. The file includes commands to delete any existing data for each section if any exists.

Note: Unlike `write_floorplan`, the `write_floorplan_script` command does not include special-routing like P/G routing. You should use `write_def_by_section -special_net_routing` for that purpose.

Note: The `create_bus_guide -type {hard | soft}` command supports soft/hard constraints to guide Innovus routers. With the hard constraint type, after running Early Global Route (eGR), NanoRoute (NR), or NanoRoute High Frequency Router (NRHF), the route obeys the path of the bus guide but with soft constraints the bus guide only guides the route path. The tool can route the net out of the bus guide. The `write_floorplan_script` command writes out these constraints types to the output file.

Parameters

`-append_to_file`	Appends new data to the file, by default it will write out to the file specified by out_file. *Data_type*: bool, optional
`-compact_row`	When specified, it enables the `write_floorplan_script` command to write out the row-creation commands in a simple and compact way. *Data_type*: bool, optional
out_file	Writes the floorplan Tcl commands to the specified file. *Data_type*: string, required
`-help`	Outputs a brief description that includes the type and default information for each `write_floorplan_script` parameter. For a detailed description of the command and all of its parameters, use the man command: `man write_floorplan_script`
`-no_snap_to_grid`	Specifies that the object should not be snapped to the specified grid and keep the location of object the same before and after loading the dumped script. *Data_type*: bool, optional
`-sections {boundary row place_blockages route_blockages pin_blockages groups constraints pins iopad areaio bump partitions blocks global_net_connect bus_sink_groups black_boxes net_group_and_bus_guide relative_fplan pin_groups pin_guides custom_shape cover_cell}`	

	Specifies one or more specific floorplan sections to write out to the file. • `areaio`: Writes out areaio information. • `black_boxes`: Writes out location, shape, and orientation of black box instances. • `blocks`: Writes out placement and orientation of all blocks. • `boundary`: Writes out the floorplan boundary, including the boundary to IO and core box spacing, and the global row parameters required by the `floorPlan` command to regenerate the rows automatically. **Note**:The `floorPlan` command will auto-generate the rows also, unless "rows" is included with `-sections` to override them with the current rows. • `bump`: Write out all the bumps. • `bus_sink_groups`: Writes bus sink group information. • `constraints`: Writes out any region, fence, and guide constraints for groups or hinsts. • `cover_cell`: Writes out cover cell information. • `custom_shape`: Writes out custom shape information. • `global_net_connect`: Writes out all the global net connections. • `groups`: Writes out any group definitions (group names and list of instances). It also writes out the groups belonging to power domains. • `iopad`: Writes out all the IO pads. • `net_group_and_bus_guide`: Writes out all the net groups and bus guides. • `partitions`: Writes out all the partitions. • `pin_blockages`: Writes out all the pin blockages. • `pin_groups`: Writes out all the pin groups. • `pin_guides`: Writes out all the pin guides. • `pins`: Writes out all the external IO pins. • `place_blockages`: Writes out placement/density blockages. • `relative_fplan`: Writes out relative floorplan constraints and all the placement relationship of floorplan objects independently from the actual coordinates in a floorplan. **Note**: You can use `write_floorplan_script -sections relative_fplan` to save relative floorplan info. • `route_blockages`: Writes out routing blockages. • `row`: Writes out all the core and IO rows. *Data_type*: (enum)+, optional
`-selected`	Specifies the floorplan object to write out. **Note:** The tool does not support to write out standard cells. *Data_type*: bool, optional

Example

- The following command writes out the user specified (row) section to the `write_floorplan_script.tcl` file.

```
write_floorplan_script -sections {row} out_file write_floorplan_script.tcl
```

In write_floorplan_script.tcl, something like the following will be written out:

```
# Standard Cell Row
delete_row -all
set_db current_design .row_spacing_type 2
set_db current_design .row_spacing 0
create_row -site tsm3site -area 335.280 335.440 1119.360 340.483 -flip_first_row -no_check
create_row -site tsm3site -area 335.280 340.483 1119.360 345.526 -no_check
create_row -site tsm3site -area 335.280 345.526 1119.360 350.569 -flip_first_row -no_check
create_row -site tsm3site -area 335.280 350.569 1119.360 355.612 -no_check
```

- The following command by default writes out all the sections to the `write_floorplan_script.tcl` file

  ```
  write_floorplan_script out_file write_floorplan_script.tcl
  ```

  In `write_floorplan_script.tcl`, something like the following will be written out:

  ```
  # floorplan boundary
  set ret [catch {create_floorplan -no_resize -no_snap_to_grid -box_size 0.000 0.000 1454.740 1447.062
  235.000 235.000 1219.740 1212.062 335.280 335.440 1119.740 1112.062}]
  if {$ret != 0} {
  create_floorplan -no_snap_to_grid -box_size 0.000 0.000 1454.740 1447.062 235.000 235.000 1219.740
  1212.062 335.280 335.440 1119.740 1112.062}
  # Standard Cell Row
  delete_row -all
  set_db current_design .row_spacing_type 2
  set_db current_design .row_spacing 0
  create_row -site tsm3site -area 335.280 335.440 1119.360 340.483 -flip_first_row -no_check
  create_row -site tsm3site -area 335.280 340.483 1119.360 345.526 -no_check
  create_row -site tsm3site -area 335.280 345.526 1119.360 350.569 -flip_first_row -no_check
  create_row -site tsm3site -area 335.280 350.569 1119.360 355.612 -no_check
  # netGroupAndBusGuide
  # Global Net Connection
  delete_global_net_connections
  commit_global_net_rules
  # partition
  # Placement obstruction
  delete_obj [get_db place_blockages -if { .name == *** | .type == *** } ]
  # Routing obstruction
  delete_route_blockages -all
  # Pin blockage
  delete_route_blockages -all
  # Instance group
  # Hierarchical Instance's constraint(Fence, Region or Guide)
  # Pin
  # IO pad
  set inst [get_db insts IOPADS_INST/Pibiasip]
  set_db $inst  .location 0 548.16
  set_db $inst  .orient  R270
  set_db $inst  .place_status placedset_db [get_db insts IOPADS_INST/Pibiasip ] .place_status fixed
  # Area IO
  # Bump
  ```

- The following command writes out the user specified (custom_shape) section to the `write_floorplan_script.tcl` file

  ```
  write_floorplan_script -sections {custom_shape} out_file write_floorplan_script.tcl
  ```

  In `write_floorplan_script.tcl`, something like the following will be written out:

  ```
  # custom shapes
  delete_gui_object -shape
  add_gui_shape -layer <layer>  -line {x1 y1 x2 y2}
  add_gui_shape -layer <layer>  -polygon {x1 y1 x2 y2 …}
  ```

Related Information

- `write_floorplan`
- `floorplan` Category Attributes
- Floorplanning the Design chapter in the *Innovus Stylus Common UI User Guide*
  - Saving and Loading Floorplan Data

write_io_file

```
write_io_file
[-help]
[-version_2]
[-locations | -by_order]
[-include_cell_name]
[-only_selected_bump]
[-template [-io_order {default | clockwise | counterclockwise}]]
[-relative_orient]
out_file
```

Saves the current I/O information to a Version: 3 file by default. It also provides you the option to save the I/O information in a Version: 2 file for 6.2 and previous release of the software. The saved file can be used to specify the I/O Assignment File in the Design Import form. This is used to save the I/O pin or pad order to a file after assigning pins or pads by loading a floorplan file or DEF data. This I/O Assignment file is rule-based, and the keyword in the I/O file is Version: 3. You can use this command after floorplanning the design.

Note: The `write_io_file` command saves the bump connect target information in the I/O file by default.

Note: The `write_io_file` command will write all the information of I/O pins and I/O pads if the design has both IO pins and IO pads.

Parameters

`-help`	Prints a brief description that includes the type and default information for each `write_io_file` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man write_io_file`
`-by_order`	Specifies the order of the I/O pad placement locations in the I/O file. *Data_type*: bool, optional
`-include_cell_name`	Specifies that the cell name of the I/O instance must be included in the I/O file. *Data_type*: bool, optional
`-io_order {default | clockwise | counterclockwise}`	
	Specifies the order of placing the I/O pins in the I/O template file. You can choose one of the following: • `default`: Specifies the default pin order. For a vertical edge, the default pin order is from the bottom to the top. For a horizontal edge, the default pin order is from the left of the right. • `clockwise`: Specifies the pin order in the clockwise direction. • `counterclockwise`: Specifies the pin order in the clockwise direction. **Note:** For I/O cells, the write_io_file command will write out only the default I/O order. *Data_type*: string, optional
out_file	Specifies the name of the pin order file to be written to. *Data_type*: string, required
`-locations`	Generates an I/O file that has placement locations rather than a sequence or order of the I/O pads. Placement co-ordinates are relative to the design boundary and are not absolute locations. *Default:* If you do not specify this parameter, it generates an order file. *Data_type*: bool, optional
`-only_selected_bump`	Saves only the selected bumps to the IO file. By default, `write_io_file` dumps out embedded bumps to the IO file. When the `-only_selected_bump` option is specified, only selected bumps are dumped out and embedded bumps are ignored. *Data_type*: bool, optional
`-relative_orient`	Specifies an I/O orientation relative to the bottom (south) side. *Data_type*: bool, optional
`-template`	Performs initial I/O placement by randomly distributing the I/Os evenly. Writes out an I/O template file with evenly spaced I/O pins, after design import. Once the I/O template file is generated, you can modify the I/O file manually and reload the file in Innovus. *Data_type*: bool, optional
`-version_2`	Saves the current I/O information to a Version: 2 file for 6.2 and previous release of the software. If you specify this parameter for saving an I/O file consisting of rectilinear design for I/O pads, Innovus displays an error message. This is because the rectilinear design for I/O pads is supported only in the current release version of Innovus. *Default*: If you do not specify this parameter, the I/O information is saved in a Version: 3 file. *Data_type*: bool, optional

Example

- The following command writes the current I/O pad and pin information to Version: 3 file:
  ```
  write_io_file myIoPlan.io
  ```
- The following command writes the current I/O pad and pin information to a Version: 2 file and specifies the clockwise placement of the I/O pins in the I/O template file:
  ```
  write_io_file -version_2 -io_order clockwise -templatelate myIoPlan.io
  ```

Related Information

- `read_io_file`
- Data Preparation chapter in the *Innovus Stylus Common UI User Guide*
  - Generating the I/O Assignment File

write_macro_place_constraint

```
write_macro_place_constraint
[-help]
[-cpg_scope {selected macro_only all}]
-out_file string
[-sections {array basic cpg pg_model_over_macros pg_resource_model}]
```

Enables you to write constraints into a TCL format file from the database.

Parameters

`-cpg_scope {selected macro_only all}`	
	Specifies the instances that are to be written in the CPG guide section. *Data_type*: enum, optional
`-help`	Outputs a brief description that includes the type and default information for each `write_macro_place_constraint` parameter. For a detailed description of the command and all of its parameters, use the man command: `man write_macro_place_constraint`
`-out_file string`	Specifies the name of the file to which the macro place constraints are written. *Data_type*: string, required
`-sections {array basic cpg pg_model_over_macros pg_resource_model}`	
	Specifies the constraint sections to be written. • `array`: Writes macro array constraints. • `basic`: Writes non-default macro place constraints set by the `set_macro_place_constraint` command. • `cpg`: Writes out the guide file for initializing the orientation of the instances and placing them close to the specified location. • `pg_model_over_macros`: Writes PG modeling for macros. • `pg_resource_model`: Writes PG resource modeling for standard cell areas. **Note**: This option is used to model PG over the whole design if the `-pg_model_over_macros` option is not specified. **Note:** Only the basic constraints are dumped out if the -sections option is not specified. *Data_type*: (enum)+, optional

Example

- The following command writes the basic macro place constraints to the xyz file:
  ```
  write_macro_place_constraint -cpg_score all -out_file xyz
  ```

Related Information

- `check_macro_place_constraint`
- `create_pg_model_for_macro_place`
- `get_macro_place_constraint`
- `reset_macro_place_constraint`
- `set_macro_place_constraint`
- Floorplanning the Design chapter in the *Innovus Stylus Common UI User Guide*
- Using the Mixed Placer chapter in the *Innovus Stylus Common UI User Guide*

write_route_rule

```
write_route_rule
[-help]
ndr_rule_name
{-def_file def_file_name | -lef_file lef_file_name}
```

Exports/saves the new rule that was created with the `add_ndr` command into a LEF/DEF file. You can use this command after creating a non-default rule.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `write_route_rule` parameter. For a detailed description of the command and all of its parameters, use the man command: `man write_route_rule`
ndr_rule_name	Specifies the name of a non-default rule. *Data_type*: string, required
`-def_file` *def_file_name*	
	Specifies the name of a DEF file, to which the non-default rule is exported. *Data_type*: string, required
`-lef_file` *lef_file_name*	
	Specifies the name of a LEF file, to which the non-default rule is exported. *Data_type*: string, required

Example

The following command exports the non-default rule, testndr to the LEF file, all.lef.

```
write_route_rule testndr -lef_file all.lef
```

Related Information

- `add_ndr`
- Using the NanoRoute Router chapter in the *Innovus Stylus Common UI User Guide*
  - Using Non-Default Rules
- Floorplanning the Design chapter in the *Innovus Stylus Common UI User Guide*

12

Flowkit Commands and Attributes

- check_flow
- create_flow
- create_flow_step
- define_feature
- delete_flow_config
- edit_flow
- end_flow
- flow Category Attributes
- flowtool Category Attributes
- get_feature
- get_flow_config
- init_flow
- is_feature
- is_flow
- read_flow
- report_flow
- reset_flow
- run_flow
- schedule_flow
- set_feature
- set_flow_config
- set_flowkit_read_db_args
- set_flowkit_write_db_args
- write_flow
- write_flow_template

check_flow

```
check_flow
[-help]
[-fail string]
[-flow string]
[-from string]
[-proc string]
[-step string]
[-to string]
```

Creates a single flow_step object and fills-in the required, and some optional attributes. Further modifications of the step can be performed using the `set_db` command. If you try to define a step with the same name as an existing step, it results in error. To redefine a flow step, you must use the `set_db` command to modify the attributes, or delete and recreate it.

Note: Unlike Tcl procedures, flow steps do not take arguments and refer to attributes instead. The use of attributes requires that the data on which a step relies, is stored in the database and may be examined later.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `check_flow` parameter. For a detailed description of the command and all its parameters, use the man command: `man check_flow`
`-fail` *string*	Reports a problem. this is not used for analysis. It can only be used from Tcl called as part of static checking. Calling from outside static checking results in error. *Data_type*: string, optional
`-flow` *string*	Runs the specified flow. the `from`/`to` values are based on the flow steps. *Default:* `flow_step_order` *Data_type*: string, optional
`-from` *string*	Specifies the name of the step to start the flow from. By default, the flow starts from the first step that has not been run, or the first step if all steps have been run previously. **Note:** This must be a step in the flow. *Data_type*: string, optional
`-proc` *string*	Checks the specified procedure instead of analyzing a flow step. *Data_type*: string, optional
`-step` *string*	Runs just the specified step. This takes priority over `from`/`to` options. *Data_type*: string, optional
`-to` *string*	Specifies the name of the step to run to (inclusive; in other words, the named step is executed). By default, runs to the end of the last step in the flow. Must be a step in the flow, later than the `-from` step (if any). *Data_type*: string, optional

Related Information

Flowkit chapter of the *Innovus Stylus Common UI Migration Guide* document.

create_flow

```
create_flow
[-help]
[steps]
-name name
[-owner string]
[-skip_metric]
[-tool tool]
[-tool_options options]
```

Creates a new, named flow object. `run_flow` then allows the execution of a named flow (in addition to its existing functionality).

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `create_flow` parameter. For a detailed description of the command and all its parameters, use the man command: `man create_flow`
`[steps]`	Specifies a list of flow step names in the order they must be executed. **Note:** This is an optional parameter. You can either provide `[steps]` or use the `-tool` parameter. *Data_type*: string, optional
`-name name`	The name of the flow object to create. *Data_type*: string, required
`-owner string`	Specifies the owner of the flow. *Data_type*: string, optional
`-skip_metric`	Restricts the tool from capturing the flow's metric snapshot. *Data_type*: bool, optional
`-tool tool`	Specifies the name of the tool with which to run this flow. This parameter allows you to create flows for other tools than the one currently running. If not specified, default is the current tool. *Data_type*: string, optional
`-tool_options options`	Specifies options for use when launching this flow. *Data_type*: string, optional

Related Information

Flowkit chapter of the *Innovus Stylus Common UI Migration Guide* document.

create_flow_step

```
create_flow_step
[-help]
body
[-categories string]
[-check checkTcl]
[-description string]
[-exclude_time_metric]
-name name
[-owner string]
[-write_db]
[-skip_metric]
```

Creates a single `flow_step` object and fills-in the required, and some optional attributes. Further modifications of the step can be performed using the `set_db` command. If you try to define a step with the same name as an existing step, it results in error. To redefine a flow step, you must use the `set_db` command to modify the attributes, or delete and recreate it.

Note: Unlike Tcl procedures, flow steps do not take arguments and refer to attributes instead. The use of attributes requires that the data on which a step relies, is stored in the database and may be examined later.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `create_flow_step` parameter. For a detailed description of the command and all its parameters, use the man command: `man create_flow_step`
`body`	Specifies a block of Tcl that is run as the main action of the step. *Data_type*: string, required
`-categories` *`string`*	Defines the metric categories to calculate the flow step. This parameter is equivalent to `create_snapshot -categories`. It indicates the metrics system to calculate and collect extra metrics above those already collected as part of the `create_snapshot` call. The possible values of this parameter include `all`, `*`, `flow`, `design`, `setup`, `power`, `check`, `clock`, `hold`, and `route`. Specify multiple categories in a list to calculate the requested set. The `all` and `*` values are default and aliases of each other. They cause the `create_snapshot` command to get the current state of the design using `[get_db [get_db current_design] .state]` and then look in `[get_db metric_category_status_map]` to assess the metric categories, which would calculate metrics. This is so that the metric category code would not try and calculate the clock category, when the design is only floor planned (for instance). *Data_type*: string, optional
`-check` *`checkTcl`*	Specifies a block of Tcl to run step-specific static checks. *Data_type*: string, optional
`-description` *`string`*	Describes the purpose for the current flow step. *Data_type*: string, optional
`-exclude_time_metric`	Ensures that the CPU and wall time of the corresponding step is not included in any parent steps. This parameter is equivalent to executing `create_snapshot -exclude_time_metric`. It indicates that the metrics system should not consider the snapshot CPU and wall time when calculating the total steps. Use it while reporting the flows so that the time spent in it is not added to the overall flow time. *Data_type*: bool, optional
`-name` *`name`*	Specifies the name of the Tcl of the step. This can consist of alphanumeric characters and underscore only, and may not start with a digit. The following names have special meanings and may not be used for steps: • next • end *Data_type*: string, required
`-owner` *`string`*	Specifies the owner of the flow. *Data_type*: string, optional
`-skip_metric`	Skips generating metrics when the step is complete. By default, every step saves updated metrics values when completed. So this parameter alters the `run_flow` command to not create a snapshot (by running the `create_snapshot` command) for that step. *Data_type*: bool, optional
`-write_db`	Automatically saves a database when the flow step is complete. *Data_type*: bool, optional

Related Information

Flowkit chapter of the *Innovus Stylus Common UI Migration Guide* document.

define_feature

```
define_feature
[-help]
feature
[-default string]
[-description string]
[-group string]
[-obj string]
[-possible_values string]
[-required]
[-type {bool string enum}]
```

Defines the feature for a flow.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `define_feature` parameter. For a detailed description of the command and all its parameters, use the man command: `man define_feature`
feature	Specifies the name of the feature. *Data_type*: string, required
`-default` *string*	Specifies the default value of the feature. *Data_type*: string, optional
`-description` *string*	Specifies the detailed description of the flow feature. If it matches multiple features, then these features are listed along with a short description for each. If it matches exactly one feature, a full description of this feature is printed. Matching no features results in an error. *Data_type*: string, optional
`-group` *string*	Outputs the exclusive group that contains features. *Data_type*: string, optional
`-obj` *string*	Specifies the flow or flow_step for a flow specific feature. The features can be limited to a flow/step, or be global and usable in all the flows/steps. *Data_type*: string, optional
`-possible_values` *string*	Specifies the list of values that are valid for the features. *Data_type*: string, optional
`-required`	Specifies that the mentioned feature is mandatory. *Data_type*: bool, optional
`-type {bool string enum}`	Specifies the type of feature value. *Data_type*: enum, optional

Related Information

Flowkit chapter of the *Stylus Common UI Migration Guide* document.

delete_flow_config

```
delete_flow_config
[-help]
key ...
```

Deletes the configuration value at the specified key, allowing the user to change the type of a `setup.yaml` key by first deleting it.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `delete_flow_config` parameter. For a detailed description of the command and all its parameters, use the man command: `man delete_flow_config`
key ...	Specifies the key to delete the configuration value. *Data_type*: string, required

Related information

- `set_flow_config`
- The Flowkit chapter of the *Stylus Common UI Migration Guide* document.

edit_flow

```
edit_flow
[-help]
[-after string]
[-append string]
[-before string]
[-enabled_feature string]
[-owner string]
[-prepend string]
[-reset]
[-unique]
```

Allows you to insert flow steps into an existing flow without redefining the flow itself. This command is very useful for flow customization.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `edit_flow` parameter.
`-after string`	Adds the specified `flow` and/or `flow_step` objects after the flow supporting a full canonical path to a flow step object. *Data_type*: string, optional
`-append string`	Appends the specified `flow` and/or `flow_step` objects before or after the flow. *Data_type*: string, optional
`-before string`	Adds the specified `flow` and/or `flow_step` objects before the flow supporting a full canonical path to a flow step object. *Data_type*: string, optional
`-enabled_feature string`	Executes the `edit_flow` command only if the specified user-defined features are enabled. *Data_type*: string, optional
`-owner string`	Specifies the owner of the flow being edited. *Data_type*: string, optional
`-prepend string`	Adds the specified `flow` and/or `flow_step` objects before or after the flow. *Data_type*: string, optional
`-reset`	Resets the customization for the specified `flow` and/or `flow_step` objects before or after the flow. *Data_type*: bool, optional
`-unique`	Prevents the addition of the flow and/or `flow_step` objects, if those are already provided. This avoids duplication of the flow defined in the YAML file and the `edit_flow` statements scripted in the `tech_flow_steps` file. **Note:** This parameter prevents the specified `flow_step` from getting added to the same flow twice, irrespective of the values provided to the `-after` or `-before` parameters. For example, if `edit_flow -append bob -unique` is called twice to insert different `flow_steps` in the floorplan flow, it will add `bob` only once. *Data_type*: bool, optional

Related Information

Flowkit chapter of the *Innovus Stylus Common UI Migration Guide* document.

Examples

- Following is an example of customizing the reference flow process using the `flow_config.template` file:

```
create_flow_step -name write_sdf {
write_sdf [get_db flow_report_directory]/[get_db local_flow].sdf
}
edit_flow -after flow_step:innovus_report_late_timing -append flow_step:write_sdf
```

- Following is an example of inserting an additional `step_2` step after the `step_1` step:

```
set file_dir [file dirname [ dict get [ info frame [ info frame ] ] file ] ]
source ${file_dir}/scripts/run_flow.tcl
create_flow_step -name step_2 {
}
edit_flow -append step_2 -after step_1
```

- Following is an example of adding flow customization to the `flow_config.tcl` file:

```
create_flow_step -name output_files {
set design_name [get_db current_design .name]
write_netlist [get_db flow_report_directory]/${design_name}.v
write_def [get_db flow_report_directory]/${design_name}.def
}
edit_flow -append flow_step:output_files -after flow_step:block_finish
```

- The following commands will promote the `edit_flow` statements to the parent edited flow just before or after the specified anchor:

```
edit_flow -unique -after init_floorplan -append add_tracks_n7
edit_flow -unique -after init_floorplan -append add_endcaps_n7
edit_flow -unique -after init_innovus_user -append init_innovus_n7
```

end_flow

```
end_flow
[-help]
flow_or_step
```

Ends the specified flow or step. This command accepts a single string argument, which is a flow or step name. The specified string must match a flow or step in the path for the current step. Else, a Tcl error occurs. Flowkit/flowtool then skips all the contiguous steps to match the given flow or step name in the hierarchy.

Note: Calling `end_flow` outside `flow_step` or inside `run_flow` (for example, `init_flow plugin`, `flow_header_tcl` or as a plain command) will produce an error.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `end_flow` parameter. For a detailed description of the command and all its parameters, use the man command: `man end_flow`
`flow_or_step`	Specifies the flow or step to be ended and skipped. *Data_type*: string, required

Related Information

Flowkit chapter of the *Innovus Stylus Common UI Migration Guide* document.

flow Category Attributes

The root attributes in the `flow` category are:

- flow_branch
- flow_caller_data
- flow_db_directory
- flow_error_errorinfo
- flow_error_message
- flow_error_write_db
- flow_exclude_time_for_init_flow
- flow_exit_when_done
- flow_feature_values
- flow_features
- flow_footer_tcl
- flow_header_tcl
- flow_hier_path
- flow_history
- flow_log_directory
- flow_log_prefix_generator
- flow_mail_on_error
- flow_mail_to
- flow_metrics_file
- flow_metrics_snapshot_parent_uuid
- flow_metrics_snapshot_uuid
- flow_overwrite_db

- flow_plugin_names
- flow_plugin_steps
- flow_post_db_overwrite
- flow_remark
- flow_report_directory
- flow_run_tag
- flow_schedule
- flow_starting_db
- flow_startup_directory
- flow_status_file
- flow_step_canonical_current
- flow_step_check_tcl
- flow_step_current
- flow_step_last
- flow_step_last_msg
- flow_step_last_status
- flow_step_next
- flow_summary_tcl
- flow_template_feature_definition
- flow_template_tools
- flow_template_type
- flow_template_version
- flow_user_templates
- flow_top
- flow_verbose
- flow_working_directory
- flow_yamllint_exec

To set the value of any root attribute, use the `set_db` command:

```
set_db attribute_name value
```

To get the current value of any root attribute, use the `get_db` command:

```
get_db attribute_name
```

To get the current value of all attributes in the `flow` category, use the following command:

```
get_db -category flow
```

flow_branch

`flow_branch string`

Default: ""

Data_type: string, read/write

Specifies the branch being run for a hierarchical flow.

flow_caller_data

`flow_caller_data string`

Default: ""

Data_type: string, read/write

Specifies the data used by the caller of the tool for identifying the flow.

flow_db_directory

`flow_db_directory` *`string`*

Default: `flow.database`

Data_type: string, read/write

Specifies the flow directory where results databases are stored.

flow_error_errorinfo

`flow_error_errorinfo` *`string`*

Default: ""

Data_type: string, read/write

Specifies the TCL error stack in the error database.

flow_error_message

`flow_error_message` *`string`*

Default: ""

Data_type: string, read/write

Specifies the TCL error message in the error database.

flow_error_write_db

`flow_error_write_db {true | false}`

Default: `true`

Data_type: bool, read/write

Runs the database when an error occurs in a flow step.

flow_exclude_time_for_init_flow

`flow_exclude_time_for_init_flow {true | false}`

Default: `false`

Data_type: bool, read/write

Marks all those snapshots as `exclude_time_metric`, which were created after running the `init_flow` command. This marking adds user steps to the `init_flow` internal plugin locations.

flow_exit_when_done

`flow_exit_when_done {true | false}`

Default: `false`

Data_type: bool, read/write

Exits the flow after running the final step.

flow_feature_values

`flow_feature_values` *`string`*

Default: ""

Data_type: string, read/write

Specifies the global feature settings.

flow_features

`flow_features` *string*

Default: ""

Data_type: string, read/write

Specifies the global feature definitions.

flow_footer_tcl

`flow_footer_tcl` *string*

Default: ""

Data_type: string, read/write

Specifies the TCL script to run when a flow ends.

flow_header_tcl

`flow_header_tcl` *string*

Default: ""

Data_type: string, read/write

Specifies the TCL script to run when a flow starts.

flow_hier_path

`flow_hier_path` *string*

Default: ""

Data_type: string, read/write

Displays the path of the current flow within the flow hierarchy.

flow_history

`flow_history` *string*

Default: ""

Data_type: string, read/write

Displays the complete flow run history.

flow_log_directory

`flow_log_directory` *string*

Default: `flow.log`

Data_type: string, read/write

Specifies the flow directory where log files are stored.

flow_log_prefix_generator

`flow_log_prefix_generator` *string*

Default: `{`

```
set logPrefix ""
# Work out the subset of steps we are running
```

```
set startPath [split $start_step "."]

set endPath [split $end_step "."]
for {set i 0} {$i < [llength $startPath]} {incr i} {
if {[lindex $startPath $i]
ne [lindex $endPath $i]} {
if {$subflow_start_step && $subflow_end_step} {
set logPrefix [join [lrange
$startPath 0 [expr $i - 1]] "."]
} else {
set logPrefix $start_step
if {$i < [llength
$endPath]} {
set logPrefix "${logPrefix}-[join [lrange $endPath $i end] "."]"
}
}

break
}
}
# Add the branch name
if {$branch ne {}} {
if {$logPrefix eq {}} {
set
logPrefix $branch
} else {
set logPrefix "$branch.$logPrefix"
}
}
# Add the top-level flow
to the prefix
if {$logPrefix eq {}} {
set logPrefix [string range $flow 5 end]
} else {
set logPrefix
"[string range $flow 5 end].$logPrefix"
}
# Fall back to tool name
if {$logPrefix eq {}} {
set
logPrefix $tool
}
# Add the flow log directory
set logPrefix [file join $flow_log_directory $logPrefix]

return $logPrefix
}
```

Data_type: string, read/write

Specifies the TCL script to create log filenames in flowtool.

flow_mail_on_error

```
flow_mail_on_error {true | false}
```

Default: `false`

Data_type: bool, read/write

Sends an email to 'flow_mail_to', if an error is detected.

flow_mail_to

```
flow_mail_to string
```

Default: ""

Data_type: string, read/write

Specifies the email address where results are sent.

flow_metrics_file

`flow_metrics_file` *string*

Default: ""

Data_type: string, read/write

Specifies the flow metrics file for reporting results.

flow_metrics_snapshot_parent_uuid

`flow_metrics_snapshot_parent_uuid` *string*

Default: ""

Data_type: string, read/write

Specifies the snapshot uuid to which the results from this flow will be appended.

flow_metrics_snapshot_uuid

`flow_metrics_snapshot_uuid` *string*

Default: ""

Data_type: string, read/write

Specifies the snapshot uuid of the most recent flow step executed.

flow_overwrite_db

`flow_overwrite_db {true | false}`

Default: `false`

Data_type: bool, read/write

Enables overwriting databases while saving.

flow_plugin_names

`flow_plugin_names` *string*

Default: *list*

Data_type: string, read/write

Displays the names of all the internal plugin points.

flow_plugin_steps

`flow_plugin_steps` *string*

Default: *""*

Data_type: string, read/write

Displays the list of plugin steps that have been added through `edit_flow`.

flow_post_db_overwrite

`flow_post_db_overwrite` *string*

Default: *""*

Data_type: string, read/write

When this attribute is set within `skip_db flow_step`, you can identify the name and type of database from the current to future flows as `flow_starting_db`.

flow_remark

`flow_remark` *`string`*

Default: *`""`*

Data_type: string, read/write

Displays the remarks from the last loaded `flow.yaml` file.

flow_report_directory

`flow_report_directory` *`string`*

Default: `flow.report`

Data_type: string, read/write

Specifies the flow directory where reports are written.

flow_run_tag

`flow_run_tag` *`string`*

Default: ""

Data_type: string, read/write

Specifies the tags for the particular flow run.

flow_schedule

`flow_schedule` *`string`*

Default: ""

Data_type: string, read/write

Specifies the set of flows to execute after the current flow completes.

flow_starting_db

`flow_starting_db` *`string`*

Default: ""

Data_type: string, read/write

Specifies the starting database for the flow run.

flow_startup_directory

`flow_startup_directory` *`string`*

Default: `/home/user_id`

Data_type: string, read/write

Specifies the directory from where the software was run.

flow_status_file

`flow_status_file` *`string`*

Default: ""

Data_type: string, read/write

Specifies the file for flow status.

flow_step_canonical_current

`flow_step_canonical_current` *string*

Default: ""

Data_type: string, read/write

Specifies the currently running canonical flow step.

flow_step_check_tcl

`flow_step_check_tcl` *string*

Default: ""

Data_type: string, read/write

Specifies the procedure to check the steps.

flow_step_current

`flow_step_current` *string*

Default: ""

Data_type: string, read/write

Specifies the currently running flow step.

flow_step_last

`flow_step_last` *string*

Default: ""

Data_type: string, read/write

Specifies the last flow step to be completed.

flow_step_last_msg

`flow_step_last_msg` *string*

Default: ""

Data_type: string, read/write

Specifies the message provided for the last step that was run.

flow_step_last_status

`flow_step_last_status` *string*

Default: `not_run`

Data_type: string, read/write

Specifies the status for the last flow step run.

flow_step_next

`flow_step_next` *string*

Default: ""

Data_type: string, read/write

Specifies the next step to run in the flow.

flow_summary_tcl

`flow_summary_tcl` *string*

Default: `{puts [report_metric -format text]}`

Data_type: string, read/write

Specifies the Tcl script to run at the end of flowtool.

flow_template_feature_definition

`flow_template_feature_definition` *string*

Default: ""

Data_type: string, read/write

Features list and status for the current template.

flow_template_tools

`flow_template_tools` *string*

Default: ""

Data_type: string, read/write

Specifies the list of tools included in the flow templates being run.

flow_template_type

`flow_template_type` *string*

Default: ""

Data_type: string, read/write

Specifies the type of template being run.

flow_template_version

`flow_template_version` *string*

Default: ""

Data_type: string, read/write

Specifies the version of the template being run.

flow_user_templates

`flow_user_templates` *string*

Default: ""

Data_type: string, read/write

Specifies the flow user template definitions.

flow_top

`flow_top` *string*

Default: ""

Data_type: string, read/write

Specifies the flow to be executed by default.

flow_verbose

```
flow_verbose {true | false}
```

Default: `true`

Data_type: bool, read/write

When set to `true`, each Tcl command inside a flow script will be logged. For example, the command executed inside the flow script will be prefixed with `@flow`:

```
@flow 1: set_db design_process_node 28
```

flow_working_directory

```
flow_working_directory string
```

Default: "."

Data_type: string, read/write

Specifies the flow directory where the flow is being run.

flow_yamllint_exec

```
flow_yamllint_exec string
```

Default: `yamllint`

Data_type: string, read/write

Specifies where `yamllint` can be found to check the `yaml` syntax.

flowtool Category Attributes

The root attributes in the `flowtool` category are:

- flowtool_exit_timeout
- flowtool_extra_arguments
- flowtool_metrics_qor_html
- flowtool_metrics_qor_excel
- flowtool_metrics_qor_text
- flowtool_metrics_qor_vivid
- flowtool_predict_full_names
- flowtool_summary_tcl

To set the value of any root attribute, use the `set_db` command:

```
set_db attribute_name value
```

To get the current value of any root attribute, use the `get_db` command:

```
get_db attribute_name
```

To get the current value of all attributes in the `flowtool` category, use the following command:

```
get_db -category flowtool
```

flowtool_exit_timeout

`flowtool_exit_timeout` *int*

Default: `30`

Data_type: int, read/write

Specifies the maximum time (in seconds) for which flowtool will wait to complete post exiting from the tool.

flowtool_extra_arguments

`flowtool_extra_arguments` *string*

Default: `""`

Data_type: string, read/write

Specifies the extra arguments passed to flowtool when running in the tool mode..

flowtool_metrics_qor_html

`flowtool_metrics_qor_html` *string*

Default: ""

Data_type: string, read/write

Specifies the metrics file in the HTML format to write after tool's exit.

flowtool_metrics_qor_excel

`flowtool_metrics_qor_excel` *string*

Default: ""

Data_type: string, read/write

Specifies the metrics file in the Excel format to write after tool's exit.

flowtool_metrics_qor_text

`flowtool_metrics_qor_text` *`string`*

Default: ""

Data_type: string, read/write

Specifies the metrics file in the txt format to write after tool's exit.

flowtool_metrics_qor_vivid

`flowtool_metrics_qor_vivid` *`string`*

Default: ""

Data_type: string, read/write

Specifies the metrics file to write after the tool's exit.

flowtool_predict_full_names

`flowtool_predict_full_names {true | false}`

Default: `false`

Data_type: bool, read/write

Shows the `skip_metric` flow name in prediction.

flowtool_summary_tcl

`flowtool_summary_tcl` *`string`*

Default: ""

Data_type: string, read/write

Specifies the TCL script to run during run_flow summary.

get_feature

```
get_feature
[-help]
[feature]
[-obj string]
```

Returns the feature for a flow.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `get_feature` parameter. For a detailed description of the command and all its parameters, use the man command: `man get_feature`
`feature`	Specifies the name of the feature. *Data_type*: string, optional
`-obj` *`string`*	Specifies the `flow` or `flow_step` to be returned. *Data_type*: string, optional

Related Information

Flowkit chapter of the *Stylus Common UI Migration Guide* document.

get_flow_config

```
get_flow_config
[-help]
key ...
[-quiet]
```

Returns the feature configuration for a flow.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `get_flow_config` parameter. For a detailed description of the command and all its parameters, use the man command: `man get_flow_config`
`key ...`	Specifies the key to get configuration value. *Data_type*: string, required
`-quiet`	Suppresses errors in case the key does not exist. *Data_type*: bool, optional

Examples

The below example presents the specification of a flow config key and use of the `get_flow_config` command to retrieve the configuration value:

In `setup.yaml`:

```
design_name: dtmf_recvr_core
```

Use the following command to access the content from tool prompt or script:

```
innovus> get_flow_config design_name
dtmf_recvr_core
```

Some `design_yaml` values can be hierarchical in nature. To retrieve such nested values, specify the keys as a sequence, as mentioned below:

In `setup.yaml`:

```
library_sets:
fast:
library_files:
- /path/to/library/data/timing/pllclk_fast.ldb
- /path/to/library/data/timing/ram_fast.ldb
- /path/to/library/data/timing/rom_fast.ldb
```

Use the following command to access the content from tool prompt or script:

```
innovus > get_flow_config library_sets fast
/path/to/library/data /timing/pllclk_fast.ldb /path/to/library/data]/timing/ram_fast.ldb /path/to/library/data /timing/rom_fast.ldb
```

This command can also be used to set particular values based on other defined entries. This requires proper quoting in the YAML files, since the “[]” characters are native operators. See the commands below for understanding:

In `setup.yaml`:

```
lib_dir: /path/to/library/data
```

```
library_sets:
fast:
library_files:
- '[get_db lib_dir]/timing/pllclk_fast.ldb'
- '[get_db lib_dir]/timing/ram_fast.ldb'
- '[get_db lib_dir]/timing/rom_fast.ldb'
```

Use the `-quiet` parameter to check if a key exists and to prevent the following message:

```
innovus > get_flow_config my_var
Flow setup value does not exist: my_var
```

```
innovus > get_flow_config -quiet my_var
```

Related Information

Flowkit chapter of the *Stylus Common UI Migration Guide* document.

init_flow

```
init_flow
[-help]
```

The command processes flow information provided by flowtool to setup and run a provided flow sequence. It outputs the affected flow root attributes, such as:

- init_flow summary
- Flow script: /grid/tfo/vol102/tiqa/flow/demo_dtmf/scripts/run_flow.tcl
- Flow: flow:block
- From: prects.block_start
- To: prects.block_finish
- Working directory: /grid/tfo/vol102/tiqa/flow/demo_dtmf
- Starting database: /grid/tfo/vol102/tiqa/flow/demo_dtmf/dbs/block.fplan.block_finish.enc
- Run tag:
- Branch name:
- Metrics file: /grid/tfo/vol102/tiqa/flow/demo_dtmf/flow.metrics
- Status file: /grid/tfo/vol102/tiqa/flow/demo_dtmf/flow.status

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `init_flow` parameter. For a detailed description of the command and all its parameters, use the man command: `man init_flow`

Related Information:

Flowkit chapter of the *Innovus Stylus Common UI Migration Guide* document.

is_feature

```
is_feature
[-help]
[feature_name]
[-obj string]
```

Tests if the specified flow feature is defined. The format of this command is similar to that of the `get_feature` command.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `is_feature` parameter. For a detailed description of the command and all of its parameters, use the man command: `man is_feature`
feature_name	Specifies the name of the feature to be tested. *Data_type*: string, optional
`-obj` *string*	Specifies if the `flow` or `flow_step` will be returned. *Data_type*: string, optional

Related Information:

- get_feature
- Flowkit chapter of the *Stylus Common UI Migration Guide* document.

is_flow

```
is_flow
[-help]
[-after string]
[-before string]
[-hierarchical]
[-inside string]
[-quiet]
```

Allows you to query during the execution of a flow for further action.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `is_flow` parameter.
`-after string`	Specifies the current execution after the `flow` or `flow_step` object. *Data_type*: string, optional
`-before string`	Specifies the current execution before the `flow` or `flow_step` object. *Data_type*: string, optional
`-hierarchical`	Displays all parent flows. *Data_type*: bool, optional
`-inside string`	Specifies the current execution within the `flow` object. *Data_type*: string, optional
`-quiet`	Ensures that errors are not reported for missing `flow` or `flow_step` objects. *Data_type*: bool, optional

Related Information:

Flowkit chapter of the *Innovus Stylus Common UI Migration Guide* document.

read_flow

```
read_flow
[-help]
filename
[-no_check]
```

Loads the content for the current flow and debugs the flow in the tool. The command accepts single argument as the name of the files that describe the `flow` and `flow_step` objects, as shown below:

```
read_flow scripts/run_flow.tcl
```

```
read_flow scripts/flow.yaml
```

This command should not be used for YAML or TCL files, in general. For example, the following command should not be used, in which the `setup.yaml` file is used by the flow, but does not contain flow information:

```
read_flow scripts/setup.yaml
```

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `read_flow` parameter. For a detailed description of the command and all its parameters, use the `man` command: `man read_flow`
`filename`	Specifies the flow script to read. *Data_type*: string, required
`-no_check`	Skips running the `yamllint` and `check_tcl` flow checks. *Data_type*: bool, optional

Related Information

Flowkit chapter of the *Innovus Stylus Common UI Migration Guide* document.

report_flow

```
report_flow
[-help]
```

Generates a report with the following information:

- The flow type, section list and generation date of the templates
- The flow step order
- One line for every item in flow_history , indicating the success of that item and the configuration of optional flow features, and any error message

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `report_flow` parameter. For a detailed description of the command and all its parameters, use the man command: `man report_flow`

Example

```
Flow Features: low_power
   (optional): prefer_area ccopt
Flow Generated: 2013-05-14 14:40 GMT
Flow Step Order:
  init place prects cts postcts postcts_hold signoff

Step       Status     prefer_area    ccopt    Message
-------------------------------------------------------------------------------
init       success    on             off
place      success    on             off
prects     success    on             off
cts        error      on             off      "Prefer_area" doesn't allow CTS.
cts        success    off            off
```

This report indicates:

- The flow was generated with the `low_power` feature enabled, and the `prefer_area` and `ccopt` features optional.
- The init, place and prects steps were run with `prefer_area` enabled, ccopt disabled, and succeeded.
- The cts step was then run with `prefer_area` enabled, `ccopt` disabled, and it failed.
- The cts step was then run with both `prefer_area` and `ccopt` disabled, and it succeeded.

There is some overlap between `report_flow` and the summary printed at the end of `run_flow` commands. The difference is that `report_flow` reports information about the entire flow history (since the last `run_flow -reset`, if any), whereas the summary at the end of `run_flow` only includes data about the steps run by that command.

Related Information

Flowkit chapter of the *Innovus Stylus Common UI Migration Guide* document.

reset_flow

```
reset_flow
[-help]
[-all]
[-edits]
[-features]
[-flows]
[-history]
[-steps]
```

Resets the content for the current flow in the software by clearing any existing flows, steps, or features. The functionality of the `-history` parameter of this command is equivalent to `run_flow -reset`.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `reset_flow` parameter. For a detailed description of the command and all its parameters, use the `man` command: `man reset_flow`
`-all`	Resets all the flows, steps, features, edits, and history set in the software. *Data_type*: bool, optional
`-edits`	Resets all the flow edits appended through the `edit_flow` command. *Data_type*: bool, optional
`-features`	Resets all the flow features and associated values. *Data_type*: bool, optional
`-flows`	Resets all the flow objects. *Data_type*: bool, optional
`-history`	Resets the flow history and state of all the attributes (such as `flow_step_canonical_current`, `flow_history`, and `flow_top`). *Data_type*: bool, optional
`-steps`	Resets all the `flow_step` objects. *Data_type*: bool, optional

Related Information

Flowkit chapter of the *Innovus Stylus Common UI Migration Guide* document.

run_flow

```
run_flow
[-help]
[-directory string]
[-flow string]
[-from string]
[-no_check]
[-no_summary]
[-predict {summary | tcl}]
[-reset]
[-step string]
[-to string]
[-yaml string]
```

The command runs one or more flow steps by default, in the order specified in a single flow.

If more than one flow is defined, the `run_flow` command must be run with `-flow`. However, if `-step` is used, that step does not have to be part of a flow.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `run_flow` parameter. For a detailed description of the command and all its parameters, use the man command: `man run_flow`
`-directory string`	Specifies the working directory in which to run the flow. If a directory is not specified, the current directory is used. **Note:** Any relative paths in attributes such as *flow_report_directory* are relative to this directory. *Data_type*: string, optional
`-flow string`	Runs the specified flow (created with the `create_flow` command). `from`/`to` options are based on the flow steps. *Default:* `flow_step_order` *Data_type*: string, optional
`-from string`	Specifies the name of the step to start from accepting a full dotted canonical path to a flow step object. By default, flow starts from the step immediately after the last successfully run step (determined by examining the value of *flow_history*), or the first step if that attribute is not set. *Data_type*: string, optional
`-no_check`	Skips the error checks prior to running. By default, the checks are run. *Data_type*: bool, optional
`-no_summary`	Skips the full summary from printing. By default, summary is printed and emailed if `flow_mail_to` is set. *Data_type*: bool, optional
`-predict {summary | tcl}`	Displays the projected future steps to run honoring the `-step`, `-flow`, `-from`, and `-to` parameters. *Data_type*: enum, optional
`-reset`	Before running, clears the flow history. After this, the flow starts from the first step if no explicit `-from` step is specified. *Data_type*: bool, optional
`-step string`	Runs just the specified step. This takes priority over `from`/`to` options. *Data_type*: string, optional
`-to string`	Specifies the name of the step to run to (inclusive; in other words, the named step is executed) accepting a full dotted canonical path to a flow step object. By default, runs to the end of the last step in the flow. Must be a step in the flow, later than the `-from` step (if any). *Data_type*: string, optional
`-yaml string`	Specifies the yaml file to source before running the flow. *Data_type*: string, optional

Examples

- The following command prints the hierarchical list of steps to be run post the specified run:

  ```
  run_flow -from cts.block_start -to cts.schedule_report_cts -predict
  ```

  Following is the output of the above command:

  ```
  Flow steps which would run
  --------------------------
  cts.block_start
  cts.init_innovus
  cts.init_innovus.extra_settings
  cts.add_clock_spec
  cts.add_clock_tree
  cts.add_tieoffs
  cts.block_finish
  cts.schedule_report_cts
  ```

- The following command prints the list of steps as DPO names to be run post the specified run:

  ```
  run_flow -from cts.block_start -to cts.schedule_report_cts -predict -tcl
  ```

  Following is the output of the above command:

  ```
  flow_step:block_start flow_step:init_innovus flow_step:extra_settings flow_step:add_clock_spec flow_step:add_clock_tree
  flow_step:add_tieoffs flow_step:block_finish flow_step:schedule_report_cts
  ```

Related Information

Flowkit chapter of the *Innovus Stylus Common UI Migration Guide* document.

schedule_flow

```
schedule_flow
[-help]
[-branch string]
[-db string]
[-defer]
[-directory string]
[-flow string]
[-include_in_metrics]
[-no_db]
[-sync | -no_sync]
```

Schedules a flow to be run after the current flow. The details of scheduled flows are stored in the `flow_schedule` attribute.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `schedule_flow` parameter. For a detailed description of the command and all its parameters, use the man command: `man schedule_flow`
`-branch string`	Specifies the branch for the new flow. *Data_type*: string, optional
`-db string`	Specifies the starting database of the flow. If not supplied, the final database of the preceding flow (the one currently running) is used. *Data_type*: string, optional
`-defer`	Prevents the scheduled flow from running, until requested. *Data_type*: bool, optional
`-directory string`	Specifies the working directory to use for the flow. By default, the current directory is used. *Data_type*: string, optional
`-flow string`	Specifies the flow to be scheduled. *Data_type*: string, optional
`-include_in_metrics`	Includes the flow in the current metric snapshot. *Data_type*: bool, optional
`-no_db`	When specified, flowtool does not use the database saved at the end of the step as the starting database for this flow. *Data_type*: bool, optional
`-no_sync`	When specified, flowtool does not wait for this flow to complete before continuing. *Data_type*: bool, optional
`-sync`	When specified, flowtool waits for all flows in the current branch to complete before starting. *Data_type*: bool, optional

Related Information

Flowkit chapter of the *Innovus Stylus Common UI Migration Guide* document.

set_feature

```
set_feature
[-help]
[feature]
[-obj string]
-value string
```

Saves the feature for a flow.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `set_feature` parameter. For a detailed description of the command and all its parameters, use the man command: `man set_feature`
feature	Specifies the name of the feature. *Data_type*: string, optional
`-obj` *string*	Specifies the `flow` or `flow_step` to be saved. *Data_type*: string, optional
`-value` *string*	Specifies the value to set the feature. *Data_type*: string, required

Related Information

Flowkit chapter of the *Stylus Common UI Migration Guide* document.

set_flow_config

```
set_flow_config
[-help]
key
value ...
```

Saves the feature configuration for a flow.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `set_flow_config` parameter. For a detailed description of the command and all its parameters, use the man command: `man set_flow_config`
`key`	Specifies the key to set configuration value. *Data_type*: string, required
`value ...`	Specifies the value to set configuration value. *Data_type*: string, required

Related Information

- Flowkit chapter of the *Stylus Common UI Migration Guide* document.
- delete_flow_config

set_flowkit_read_db_args

```
set_flowkit_read_db_args
[-help]
[-dynamic_view analysis_view]
[-hold_views hold_views]
[-leakage_view analysis_view]
[-lef_files list_of_files]
[-setup_views setup_views]
[{-oa_lib_cell_view {libname cellname viewname}}] [{-no_timing}] [{-no_timing_graph}] [{-mmmc_file fileName}]
```

Controls the run-time options used by `read_db` when a flow script is loading `flow_starting_db` specified by `flowtool` or used by `schedule_db`.

Note: This command supports all the parameters supported by `read_db`.

Parameters

`-help`	Prints a brief description that includes the type and default information for each `set_flowkit_read_db_args` parameter. For a detailed description of the command and all its parameters, use the `man` command: `man set_flowkit_read_db_args`
`-dynamic_view` *analysis_view*	Specifies the analysis view for the dynamic power analysis. *Data_type*: string, optional
`-hold_views` *hold_views*	Specifies the hold views for loading rather than the active hold views. *Data_type*: string, optional
`-leakage_view` *analysis_view*	Specifies the analysis view for the leakage power analysis. *Data_type*:
`-lef_files` *list_of_files*	Specifies the list of LEF files for loading. Use this option if you want to override the existing LEF files in the design being restored and load different LEF files instead. • This option is intended to allow upgrading to new versions of LEF files that are compatible with the old versions. The list of all LEF files used by this design must be provided. • If any pin locations, cell boundaries, or OBS shapes are changed for used cells, new DRC violations may occur. • If pins are added or removed from used cells, or used cells are removed, fatal errors may occur while loading the netlist. • If the removed pins and cells are unused in the design, warnings or errors may still occur, if any library-specific data was stored for those cells or pins during the design flow (such as `dont_touch` data). *Data_type*: string, optional
`-mmmc_file` *fileName*	Specifies the path name to a Multi-Mode/Multi-Corner (MMMC) configuration file (for example, the same file format used by `read_mmmc`) to replace the MMMC file saved by `write_db` (and all the corresponding .lib files, .sdc files, etc. referenced by the MMMC file). *Data_type*: string, optional

`-no_timing`	Skips reading of the MMMC file, timing libraries and timing constraints. **Note:** • Most designs that use LEF rely on the .lib information to define the bus order for MACROs that have bus bit terminals. Therefore, using the -`no_timing` parameter can result in the connectivity to bus ports of leaf level instances being reversed. When `.lib` information is not available, the default is for bus ports to be treated as descending (for example, [7:0]). While using Verilog modules or OpenAccess abstract views to define the bus order, the connectivity will still be correct even when `-no_timing` is specified. • Timing libraries cannot be loaded in the same session after a design is read using the `-no_timing` option. You must use the `set_flowkit_read_db_args` command without the `-no_timing` parameter in a new session to load the design with timing libraries and constraints. • If you save a design after `set_flowkit_read_db_args -no_timing`, it will not save timing information unless you use `set_flowkit_write_db_args -add_ignored_timing`. *Data_type*: bool, optional
`-no_timing_graph`	Skips reading the timing graph even if it was saved to the disk by `set_flowkit_write_db_args`. *Data_type*: bool, optional
`-oa_lib_cell_view {`*`libname cellname viewname`*`}`	Specifies the library name, cell name, and view name for reading an OpenAccess DB design. *Data_type*: string, optional
`-setup_views` *`setup_views`*	Specifies the setup views for loading rather than the setup views active during `set_flowkit_write_db_args`. *Data_type*: string, optional

Related Information

- Flowkit chapter of the *Innovus Stylus Common UI Migration Guide* document.
- read_db

set_flowkit_write_db_args

```
set_flowkit_write_db_args
[-help]
[-add_ignored_timing]
[-def]
[-lib]
[-no_constraint]
[-no_fill]
[-no_pvs_fill]
[-no_wait string]
[-sdc]
[-user_path]
[-verilog]
[{-rc_extract}] [{-timing_graph}] [{-oa_lib_cell_view string}] [{-oa_view string}] [{-lib_to_ldb}]
```

Allows you to save databases through flowkit in the Open Access format and specify the library/cell/view information. The `set_flowkit_write_db_args` command can be used to extend the options used to save a design on flow-steps, which are marked with `skip_db false`. This command needs to be placed inside the `flowstep body_tcl` of interest since it only affects that `flowstep` and resets back to default state when that step completes. The format of this command is similar to what would be used for `write_design`:

```
create_flow_step -name save {set_flowkit_write_db_args -rc -lib2ldb}
```

This saves a database named `save`, which uses the `-rc` and `-lib2ldb` options for `write_db`. Previous and subsequent flowsteps use default `write_design` options unless overridden using the `set_flowkit_write_db_args` command.

Note: This command supports all the parameters supported by `write_db`.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `set_flowkit_write_db_args` parameter.

`-add_ignored_timing`	Copies the old timing data when loaded in physical-only mode. If you use `set_flowkit_read_db_args -no_timing` to read the DB, the timing data is not loaded. This option allows you to copy the ignored timing data from the original `read_db` location and add it back during `set_flowkit_write_db_args` like this:
	`set_flowkit_read_db_args -no_timing test1`
	`<physical editing commands ...>`
	`set_flowkit_write_db_args -add_ignored_timing test2`
	This usage model is intended to allow faster physical-only editing, like DRC fixing, where the netlist is not modified. If you change the netlist, unpredictable results may occur when reading the `test2` DB using timing data copied from `test1`.
	Data_type: bool, optional

`-def`	Saves a DEF file, in addition to the normal DB files in the DB directory. This parameter is normally not used for flat or block flows and is not used by `set_flowkit_read_db_args`. It can be useful in a hierarchical flow, where you want to use `assemble_design` to read only the netlist and the DEF files without opening up each design DB and writing out the Verilog and DEF files separately (see `-verilog`). *Data_type*: bool, optional
`-lib`	Resolves symbolic links of the timing lib files and side files. Normally external file references, such as to the LEF, .lib, SDC, files are saved using full path-name, symbolic links to the original files inside a libs sub-directory inside the DB directory. This parameter forces copies of the files rather than symbolic links in the libs sub-directory so there are no external file references. That makes the DB directory much larger, and makes it suitable for sending to a different network where the library and external data files are not available. An OA DB will still have OA library references using the normal cds.lib methodology, but any other external file references will be copied inside the cellview directory. *Data_type*: bool, optional

`-lib_to_ldb`	Converts the `.lib` timing libraries referenced in the MMMC file to the `.ldb` format and writes them in a libs sub-directory inside the output design directory. References to the original .lib files in the MMMC file are changed to the corresponding `.ldb` files inside the libs sub-directory. This makes the design directory much larger because it has all the timing library data inside it and is not recommended for normal flows. Normally, you should setup `.ldb` files in an external library directory and have the MMMC files use them directly to avoid this overhead. Using `.ldb` files will make `set_flowkit_read_db_args` faster when it reads the library data, so this option is useful for doing "throw-away" debugging experiments that require repeated sessions at the cost of a one-time conversion, and carrying the library data with the design. *Data_type*: bool, optional
`-no_constraint`	Saves the timing graph data without the SDC constraints data. This parameter requires the `-timing_graph` parameter and is useful in reducing the disk size, and speeding up `set_flowkit_read_db_args` and `set_flowkit_write_db_args`. However, the timing-graph data cannot be loaded by any other release version. Additionally, the timing constraints would be lost without the SDC files, if loaded by a different release version. **Note:** You can still load the timing-graph with the current release version, and then use `set_flowkit_write_db_args` without the `-no_constraint` parameter to regenerate the SDC files from the timing-graph and then use `set_flowkit_read_db_args` with a later release. *Data_type*: bool, optional
`-no_fill`	Prevents writing of any metal or via fill shapes inside the database. This is useful during the ECO routing, since the fill data is regenerated anyway. This way, it is not worth saving the current fill data. All the fill data that would go to DEF FILLS section (floating fill data), or the DEF SPECIALNETS section with + SHAPE FILLWIRE or FILLWIREOPC will be ignored and not written. *Data_type*: bool, optional
`-no_pvs_fill`	Prevents saving the PVS fill data file. In some PVS flows, a stream (GDS) file from PVS is saved inside the software DB directory. During `set_flowkit_read_db_args`, this file is not read, but can be optionally loaded or used later in the flow. By default, if the PVS fill file exists, `set_flowkit_write_db_args` will copy it from the `set_flowkit_read_db_args` location to the new DB directory. If ECO changes to the current design have made the PVS fill file invalid, use this parameter to stop it from being copied. *Data_type*: bool, optional
`-no_wait` *string*	Writes the design in a separate process. You can continue executing new commands simultaneously while the design is being saved. This is useful when you want to save intermediate results without waiting for the save to complete. *Data_type*: string, optional
`-oa_lib_cell_view` *string*	Saves the database to an Open Access library cell view. *Data_type*: string, optional
`-oa_view`	Specifies the Open Access view name. *Data_type*: string, optional
`-rc_extract`	Writes RC extraction data, including data for incremental extraction. By default, the software does not automatically save RC extraction data to save disk space and run-time. The `-rc_extract` parameter does not save RC extraction data if the design is not extracted or if extract_rc_engine = pre_route. *Data_type*: bool, optional
`-sdc`	Saves the timing constraint file even if it is not modified. *Data_type*: bool, optional
`-timing_graph`	Saves the timing graph data. *Data_type*: bool, optional
`-user_path`	Saves external file references exactly same as the user entered it. *Data_type*: bool, optional

`-verilog`	Saves a Verilog file inside the DB directory, in addition to the normal DB files. This parameter is not normally used for flat or block flows. It can be used in a hierarchical flow, where you want to use `assemble_design` to read only the Verilog and the DEF files without opening up each design DB and writing out the Verilog and DEF files separately. *Data_type*: bool, optional

Related Information

- `write_db`
- Flowkit chapter of the *Innovus Stylus Common UI Migration Guide* document.

write_flow

```
write_flow
[-help]
[-directory string]
-out_file string
[-flow_tcl]
[-flow_yaml]
[-include_state]
[-setup_yaml]
```

Generates a Tcl script that may be sourced to reproduce the current flow. The script includes commands to:

- Set the value of all top-level attributes with non-default values to their current values
- Recreate all flow step objects with their current attributes, aside from those relating to run history (specifically, `run_count` and `status`).

Sourcing this script reproduces (but not runs) the current flow in a fresh session, and forms a user readable and editable record of the flow (compared to the flow data saved in the DB, which should not be read or edited directly). The command also supports writing a YAML file version of the memory flow. This is an outline of the content, and may not exactly match what the `write_flow_template` command produces.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `write_flow` parameter. For a detailed description of the command and all its parameters, use the `man` command: `man write_flow`
`-directory string`	Specifies the directory is which all the files need to be written. *Data_type*: string, optional
`-flow_tcl`	Writes only the `flow.tcl` file in the directory. *Data_type*: bool, optional
`-flow_yaml`	Writes only the `flow.yaml` file in the directory. *Data_type*: bool, optional
`-include_state`	Writes out the current flow state. This can be used to restore the flow. *Data_type*: bool, optional
`-out_file string`	Specifies the name of the output file to generate. *Data_type*: string, optional
`-setup_yaml`	Writes only the `setup.yaml` file in the directory. *Data_type*: bool, optional

Related Information

Flowkit chapter of the *Innovus Stylus Common UI Migration Guide* document.

write_flow_template

```
write_flow_template
[-help]
[-describe string]
[-directory string]
[-enable_feature string]
[-ip_steps string]
[-list]
[-optional_feature string]
[-tech_steps string]
[-template_file string]
[-tools]
[-type string]
[-user_steps string]
[-version string]
```

Creates a set of template TCL files to be used as the basis for a flow. This command is used at the beginning after which, you can manage the generated and modified files. When invoking the command, you will select the type of flow required (implying a subset of template scripts to include) and the arrangement of sections across generated files.

On completion, a message is given to indicate creation of the flow template in the specified directory.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `write_flow_template` parameter. For a detailed description of the command and all its parameters, use the `man` command: `man write_flow_template`
`-describe string`	Outputs a detailed description of available flow features for flow types matching the provided string. A features table includes names and description for each flow (refer to the Examples section). Matching no flow types results in an error. *Data_type*: string, optional
`-directory string`	Specifies the name of the directory for saving the generated files. If no directory is specified, the command uses the name *flow*. If required, the tool appends "_" and an integer to the directory name to avoid overwriting an existing directory. For example, `flow_1` and `flow_2`). Default: `scripts` *Data_type: string, optional*
`-enable_feature string`	Specifies a list of flow features that are unconditionally incorporated into the generated scripts. *Data_type*: string, optional
`-ip_steps string`	Overlays the contents of the generated `flow.yaml` file and inserts the specified scripts under the `ip_flow_steps` flow. *Data_type*: string, optional
`-list`	Lists the available flow templates. *Data_type*: bool, optional
`-optional_feature string`	Specifies a list of flow features that are conditionally incorporated into the generated scripts. Both the enabled and disabled code are included, with conditionals based on a new attribute. *Data_type*: string, optional
`-tech_steps string`	Overlays the contents of the file(s) generated by running the `write_flow_template` command. The file(s) provided with this parameter are copied into the `scripts` directory inside a `tech` sub-folder and appear in `tech_flow_steps` for the stylus flows. *Data_type*: string, optional
`-template_file string`	Specifies the flow template file. *Data_type*: string, optional
`-tools`	Returns the list of tools whose scripts should be included in the flow. *Data_type*: string, optional
`-type string`	Specifies the type of flow to be used. *Data_type*: string, optional
`-user_steps string`	Overlays the contents of the generated `flow.yaml` file and inserts the specified scripts under the `user_flow_steps` flow. *Data_type*: string, optional
`-version string`	Specifies the version of the flow to be used. *Data_type*: string, optional

Related Information

Flowkit chapter of the *Innovus Stylus Common UI Migration Guide* document.

Examples

- Following is an example of fetching the detailed description of the available flow features for the `block` flow sequence using this parameter:

```
innovus> write_flow_template -describe block
Features for flow templates matching 'block'
```

```
Template block
--------------
Provider : tool
Max version : 1
Description : Standard block flow
This is the default template
+--------------------------+-------------------------------------------------------------------------+
| Feature | Description |
+--------------------------+-------------------------------------------------------------------------+
| clock_design | Run discrete clock expansion |
| dft_compressor | Add flow support for scan chains with compression insertion |
| dft_simple | Add flow support for scan chain insertion |
| express | Enable express synthesis and implementation flow |
| opt_early_cts | Implement early clock tree for use during prects optimization |
| opt_eco | Run opt_design during eco flow |
| opt_postcts_hold_disable | Disable postcts hold fixing |
| opt_postcts_split | Run postcts opt_design for setup and hold as separate steps |
| opt_postroute_split | Run postroute opt_design for setup and hold as separate steps | | opt_signoff_area | Run signoff base area
optimization as part implementation flow|
| opt_signoff_drv | Run signoff base drv optimization as part implementation flow | enabled |
| opt_signoff_dynamic | Run signoff base dynamic optimization as part implementation flow | disabled |
| opt_signoff_hold | Run signoff base hold optimization as part implementation flow | disabled |
| opt_signoff_leakage | Run signoff base leakage optimization as part implementation flow | enabled |
| opt_signoff_setup | Run signoff base setup optimization as part implementation flow |
| report_inline | Run report generation as part of parent flow versus schedule_flow |
| route_track_opt | Adds track based optimization to route_design |
| synth_physical | Full physically aware synthesis flow |
| synth_spatial | Physically aware synthesis flow with spatial final optimization |

+--------------------------+-------------------------------------------------------------------------+
```

- When one or more of these features are enabled, a new `opt_signoff` flow is added to the `block` flow sequence. Each of these features corresponds to calling the `opt_signoff` command. For example, on enabling all the `opt_signoff` features, the following flow sequence is displayed:

```
flow:block
flow:opt_signoff
flow_step:run_opt_signoff_area
opt_signoff -area
flow_step:run_opt_signoff_drv
opt_signoff -drv
flow_step:run_opt_signoff_setup
opt_signoff -setup
flow_step:run_opt_signoff_hold
opt_signoff -hold
flow_step:run_opt_signoff_leakage
opt_signoff -leakage
flow_step:run_opt_signoff_dynamic
opt_signoff -dynamic
```

- Following is an example of viewing available flow types using the `-list` parameter:

```
innovus> write_flow_template -list
tool provided templates:
+-----------+---------+--------------------------------+
| Name | Version | Description |
+-----------+---------+--------------------------------+
| block | 1 | Standard block flow (default) |
| hier | 1 | Hierarchical flow |
+-----------+---------+--------------------------------+
```

- Following is an example of generating template of `block` type in the `scripts` directory using the `-enable_feature` parameter:

```
innovus> write_flow_template \
+ -enable_feature dft_simple \
+ -tools "genus"
##FLOW_KIT: generating tool provided flow type 'block'
Flowkit template writing file scripts/run_flow.tcl
Flowkit template writing file scripts/common_steps.tcl
Flowkit template writing file scripts/genus_config.template
Flowkit template writing file scripts/design_config.template
Flowkit template writing file scripts/mmmc_config.template
```

```
Flowkit template writing file scripts/flow_config.template
Flowkit template writing file scripts/genus_steps.tcl
FLOWKIT MSG: 200 Completed template generation of type 'block' in directory 'scripts'.
```

- Following is an example of generating template of `block` type in the `scripts` directory using the `-optional_feature` parameter:

```
innovus> write_flow_template \
+ -optional_feature synth_physical \
+ -tools "genus"
##FLOW_KIT: generating tool provided flow type 'block'
Flowkit template writing file scripts/run_flow.tcl
Flowkit template writing file scripts/common_steps.tcl
Flowkit template writing file scripts/genus_config.template
Flowkit template writing file scripts/innovus_config.template
Flowkit template writing file scripts/design_config.template
Flowkit template writing file scripts/mmmc_config.template
Flowkit template writing file scripts/flow_config.template
Flowkit template writing file scripts/genus_steps.tcl
FLOWKIT MSG: 200 Completed template generation of type 'block' in directory 'scripts'.
```

13

General Commands and Attributes

- alias
- check_ccr
- check_netlist
- check_syntax
- compare_release
- convert_legacy_to_common_ui
- decrypt
- define_proc
- define_proc_arguments
- define_variables
- delete_floating_constants
- delete_dangling_ports
- deselect_obj
- encrypt
- eval_legacy
- get_all_layers
- get_common_ui_map
- get_compress_level
- get_layer_preference
- get_legacy
- get_message
- get_preference

- gui_bind_key
- gui_get_record_obj_by_info
- gui_move_cursor
- help
- is_common_ui_mode
- lminus
- man
- parse_proc_arguments
- print_message
- program Category Attributes
- puts
- redirect
- rename_obj
- report_area
- report_dangling_ports
- report_gate_count
- report_hidden_usage
- report_messages
- report_netlist_statistics
- report_obj
- report_obj Category Attributes
- report_summary
- resume
- select_obj
- set_compress_level
- set_current_design
- set_layer_preference

- set_license_check
- set_limited_access_feature
- set_log_post_cmd
- set_message
- set_preference
- set_proc_verbose
- set_window_preference
- source
- suspend
- system Category Attributes
- uniquify_filename
- update_names
- view_log
- gui_get_legend
- gui_set_legend
- read_taf
- report_disk
- time_info

alias

```
alias
[-help]
new_cmd
old_cmd
[-args {{new old} ...}]
```

Create command aliases for any Tcl command. It is a wrapper around the built-in Tcl “interp alias” command. All command aliases are public, so partial command matching and using <tab> to

complete a typed in command work on them. It also supports argument name aliases for procedures that use `define_proc_arguments`, `parse_proc_arguments`, or `define_proc` for parsing the arguments, and for application commands. The argument aliases do not work for built-in Tcl commands. The argument aliases are private, so they are not shown by `-help`, and do not support partial-option completion or matching.

You can use this command to migrate an existing Tcl script when some of the commands or argument names have changed, but are otherwise compatible.

Parameters

`-help`	Prints a brief description that includes type and default information for each `alias` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man alias`
`new_cmd`	Specifies the new command name that is an alias for the existing *`old_cmd`* name. It can be the same name as the *`old_cmd`* if you only want to create new `-arg` aliases, and do not need a new command alias. *Data_type*: string, required
`old_cmd`	Specifies the existing old command name. *Data_type*: string, required
`-args {{`*`new old`*`} …}`	Specifies a list of new and old argument names to create aliases. You can give multiple *`new`* names for the same *`old`* name. The new aliases are private, and do not show up with `-help`. You can create more than one alias for the same *`old`* name. *Data_type*: string, optional

Example

The following command adds a new command name alias `new_read_sdf` and some argument name aliases for `read_sdf`:

```
>alias new_read_sdf read_sdf -args {
```

```
    {-ignore_error -continue_on_error}
    {-summary -report_summary}
    {-sum -report_summary}
}
```

Then, the following commands

```
>new_read_sdf in.sdf -ignore_error -summary
>new_read_sdf in.sdf -ignore_error -sum
>new_read_s in.sdf -ignore_error -sum
```

would behave the same as

```
>read_sdf in.sdf -continue_on_error -report_summary
```

Note: Tcl partial command matching works if the partial command is unique as shown in the above example, but the alias argument names are private, and will not support partial option matching, or appear in the `-help` output.

Related Information

- `define_proc_arguments`
- `parse_proc_arguments`
- `define_proc`

check_ccr

```
check_ccr
[-help]
ccr_number
```

Enables users to check if a CCR (Cadence Change Request) created on their behalf is checked into the current release or not.

The command returns 1 if the specified CCR is integrated into this build and 0 if it is not.

- It includes all integrated CCRs that have the `Checked_In` or `Validated` status at the time of the build.
- It only includes master CCRs. Duplicate CCRs are not included.
- It currently lists all CCRs fixed in the current release stream or previous release. For example, checking in release version 16.22, will find all CCRs integrated into 16.10 through 16.22 that are in the current 16.22 release.

Parameters

`-help`	Prints a brief description that includes type and default information for each `check_ccr` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man check_ccr`
`ccr_number`	Specifies the CCR number to check. *Data_type*: string, required

Examples

- The following example checks if the CCR 123456 is checked into the Innovus release 18.13-s088_1.

```
innovus 1> getversion
18.13-s088_1

innovus 2> check_ccr 123456
1
```

check_netlist

```
check_netlist
[-help]
-out_file filename
[-sub_module_ports]
```

Performs several checks on the design netlist and creates a report, named `checknetlist.rpt`. This report shows the following information:

- Design Summary
- Design Statistics
- I/O Pad Summary
- Design Rule Checking
- Submodule Port Definition Checking (optional)

You can use this command after the design netlist is imported.

Parameters

`-help`	Prints a brief description that includes type and default information for each `check_netlist` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man check_netlist`
`-out_file` *filename*	Specifies the name of the file where the report will be written. If no name is specified, the default filename is `checknetlist.rpt`. *Data_type*: string, optional
`-sub_module_ports`	If this option is included, `check_netlist` checks the port definitions for any submodules. *Data_type*: bool, optional

Example

Error conditions are indicated by an asterisk (*) preceding the reported statistic, such as

```
Number of Ports Connect to Core Cells *: 57

Number of Ports Connect to multiple Pads *: 0

Number of Floating Ports *: 0
```

Warnings, or possible errors are indicated by a question mark in front of the statistic, such as

```
Number of undefined Floating Pins ?: 0
```

The following example shows the information contained in the generated by `check_netlist` report.

```
Design: TOPModule

------ Design Summary:

Total Standard Cell Number (cells) : 5908

Total Block Cell Number (cells) : 4

Total I/O Pad Cell Number (cells) : 0

Total Standard Cell Area ( um^2) : 133864.32

Total Block Cell Area ( um^2) : 366346.56

Total I/O Pad Cell Area ( um^2) : 0.00


------ Design Statistics:

Number of Instances : 5912

Number of Nets : 6217

Average number of Pins per Net : 3.69

Maximum number of Pins in Net : 541


------ I/O Pad summary

Number of Primary I/O Ports : 57

Number of Input Ports : 28

Number of Output Ports : 29
```

```
Number of Bidirectional Ports : 0

Number of Power/Ground Ports : 0

Number of Ports Connect to Core Cells *: 57

Number of Ports Connect to multiple Pads *: 0

Number of Floating Ports *: 0


------ Design Rule Checking:

Number of Output Pins connect to Power/Ground *: 0

Number of Input Floating Pins *: 1

Number of Output Floating Pins : 245

Number of InOut Floating Pins : 0

Number of UniPort Floating Pins : 0

Number of undefined Floating Pins ?: 0


Number of Multiple Driving Nets *: 1

Number of Parallel Driving Nets : 1

Number of Tristate Driving Nets : 0

Number of Input Floating Nets(No FanIn) *: 0

Number of Output Floating Nets(No FanOut) : 1

Number of Hign Fanout Nets (>50) : 3


------ Sub Module Port Definition Checking

* module arb has 3 port define error

* module accum_stat has 1 port define error

* module tdsp_core_glue has 128 port define error
```

Related Information

- read_netlist
- report_netlist_statistics
- write_netlist Category Attributes
- write_netlist

check_syntax

```
check_syntax
[-help]
[file1.tcl file2.tcl ...]
[nagelfar.tcl options]
[-options]
[-out_file fileName]
```

Does the “lint type” checking of Innovus Tcl script files.

It finds many common Tcl errors, such as misspelled commands, using a Tcl variable before setting it.

The `check_syntax` command is a wrapper around the public-domain Tcl linter named nagelfar. Nagelfar automatically loads the Innovus syntax and plugin files and is therefore aware of Innovus commands and options. Nagelfar is available from http://sourceforge.net/projects/nagelfar. A recent version of nagelfar is included in the installation hierarchy at: `<install_dir>/share/nagelfar`.

You can check the nagelfar documentation in the installation directory for details about its options, the checks it performs, and the steps to turn on or off various checks.

To run this command, you should have tclsh in your search path.

Parameters

`-help`	Prints a brief description that includes type and default information for each check_syntax parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man check_syntax`
`file1.tcl` *`file2.tcl`*	Specifies the Tcl files to be checked.
`nagelfar.tcl options`	Specifies the native nagelfar.tcl options. See the nagelfar docs in the installation directory for details.
`-options`	Calls “nagelfar.tcl –help” to list out all the legal nagelfar options. This option cannot be used with any other options. *Data_type*: bool, optional
`-out_file` *`fileName`*	Sends the output to the specified file, rather than the default of stdout. *Data_type*: string, optional

compare_release

```
compare_release
[-help]
[-out_file file_name]
{-base_version release_version | -base_file file_name}
[-target_version release_version | -target_file file_name]
```

Compares the two different releases and reports all the added, deleted, or modified commands, arguments, and root attributes. It also reports any data-type changes for argument values or root attributes and any enum value changes.

Parameters

`-help`	Prints a brief description that includes type and default information for each `compare_release` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man compare_release`
`-base_file` *file_name*	Specifies the `.info` file of the older base release. *Data_type*: string, optional
`-base_version` *release_version*	Specifies the older base release, such as 16.10-p001_1, from `<install>/share/innovus/version_info`. *Data_type*: string, optional
`-out_file` *file_name*	Specifie the output report file. *Default*: `stdout` *Data_type*: string, optional
`-target_file` *file_name*	Specifies the target release .info file to be compared with the older base release .info file. *Data_type*: string, optional

`-target_version` *release_version*	Specifies the version, such as 16.20-p002_1, which is compared to the older base release. Looks in `<install>/share/innouvs/version_info` for the correct `.info` file. *Default*: Current version *Data_type*: string, optional

Example

- The following command compares two releases by first finding the install directories, and the correct .info files to compare.

  ```
  >which innovus

  /tools/innovus18.11/bin/innovus
  ```

  The install directory above is /tools/innovus18.11 (for examples, the 18.1 ISR release).

  Suppose you also have the 18.1 production release in /tools/innovus.18.10. Find the .info files to compare:

  ```
  >ls /tools/innovus18.10/share/innovus/version_info

  innovus.18.10-p006_1.cui.info  innovus.18.10-p006_1.info

  >ls /tools/innovus18.11/share/innovus/version_info

  innovus.18.11-s080_1.cui.info  innovus.18.11-s080_1.info
  ```

  Then, run the `compare_release` command to compare either the legacy (.info.) or Common UI versions (.cui.info):

  ```
  innovus> compare_release \

  -base_file /tools/innovus18.10/share/innovus/version_info/innovus.18.10-
  p006_1.cui.info \

  -target_file /tools/innovus18.11/share/innovus/version_info/innovus.18.11-
  s080_1.cui.info \

  -out_file 18.11.diff
  ```

  The output file will show all the differences between the two releases.

Related Information

- innovus

convert_legacy_to_common_ui

```
convert_legacy_to_common_ui
[-help]
<in_dir_or_file>
<out_dir_or_file>
```

Allows usage of legacy scripts to be used in Stylus Common UI.

Parameters

`-help`	Prints out the command usage.
`<in_dir_or_file>`	Specifies the name of the legacy script or script directory name. *Data_type*: string, required
`<out_dir_or_file>`	Specifies the Stylus Common UI script or script directory name. *Data_type*: string, required

decrypt

```
decrypt
[-help]
file_name
```

Decrypts and evaluates a Tcl file that was encrypted with the encrypt command.

Parameters

`-help`	Prints out the command usage.
file_name	Specifies the name of the Tcl file to be decrypted and evaluated. *Data_type*: string, required

Related Topics

- encrypt

define_proc

```
define_proc
[-help]
proc_name
arg_list
proc_body
[-description proc_help]
[-hidden]
[-extended_help help_description]
```

This command defines a Tcl proc that uses an extended argument syntax to check for legal argument names and legal values. It is a super-set of the functionality supported by define_proc_arguments, and parse_proc_arguments and easier to use.

The arguments are parsed, and if any errors occur, an error message is written, and a TCL_ERROR is returned, so the *proc_body* is not executed. If the arguments are parsed successfully the Tcl vars are set accordingly, and the *proc_body* is executed.

It supports simple types like int, string, etc., built-in types like point, rect, and objects like inst, net, pin.

Commands defined with define_proc also have the same support as other Innovus commands, including:

- help support (e.g. "cmd -help" to see the options, and "help c*" would find the command named cmd, etc.)
- tab-completion of partial command or option names (e.g. "cmd<tab>" or "cmd -opt<tab>")
- partial argument name matching is allowed (e.g. "cmd -opt" will work if -opt only matches one option name prefix like -opt_test)
- redirect output to files or Tcl vars using > or >> (e.g. "cmd > my_file", or "cmd > &my_tcl_var")

Parameters

arg_list	Defines the argument list of the Tcl procedure in the following format for each argument: `{var default arg_name data_type flags help}` *Data_type*: string, required For example

```
{report_file report.out {-out_file file_name}
string optional {The output file for the report}
}
```

In this format:

- *var* is the local Tcl variable to assign, such as report_file in the example above. The value for the argument is assigned to this variable before the body of `define_proc` is invoked.
- *default* is the default value, such as report.out in the example above. If the argument is not given, then *var* will be assigned the *default* value. Note, unlike an argument value passed to the command, the type of the default value is not checked.
  For example, the following command would set int_var to "no_value" if -my_int is not given. Normally the default value should be a legal int value, but you can use this to check if the argument was given for an optional argument when there is no good default value.

  ```
  {int_var no_value {-my_int int} int optional {int
  value help description}}
  ```

  The default value is ignored if the argument is a required argument (see *flags* below).
- *arg_name* is the argument name with optional value help. Both named {-opt value} and unnamed {value} arguments are supported.

  If the first word starts with a dash, it is a named argument with optional help string. For example, `{-out_file file_name}` will appear in the help output as: -out_file <file_name>.

  If no help string is given like this: `{-out_file}`

  The help output will include the data_type like: -out_file <string>.

If the first word does not start with a dash, it is an unnamed argument and the first word is used as the help string. For example, `{out_file}` will appear in the help output as: `<out_file>`.

- *`data_type`* is the argument data type. Many types are supported including simple types, bool type, enum type, built-in types, and object types.
  - These simple types are supported: **int, float, double, string.** They all require an argument value that matches the type (e.g. -my_int 23, or -my_string "my test string"). **float** and **double** are both allowed, and mean the same thing. For example

    ```
    {int_var 0 {-my_int int} int optional {int value help description}}
    ```

    Note that a integer passed to a float or double type will get a .0 added to the end so it can be passed to expr and forced to use floating point arthimetic.
  - The **bool** type supports an argument with or without a value. If the *arg_name* definition has a value help string, then a bool value is required, otherwise a value is not allowed. So, a switch that takes no value would look like:

    ```
    {switch_var false -my_switch bool optional "-my_switch help description"}
    ```

    and would be used like this with no value

    ```
    -my_switch
    ```

    The switch_var will be set to true if -my_switch is given, and set to the default value of false if not given.

    A bool with a value required, would look like this

    ```
    {bool_var false {-my_bool bool} bool optional "bool with value help description"}
    ```

    and would be used like this

    ```
    -my_bool true
    ```

Any Tcl allowed bool value can be used for the input value (e.g. true, false, 1, 0, on, off, etc.) but the bool_var will always have either true, false or the *default* value if the argument is not given.

- These built-in types are supported: **point, rect.**
  - **point** is exactly two doubles. The input can be either a single list or list of lists style, so both {1.0 2.0} or {{1.0 2.0}} are allowed as input. The point_var will always be assigned as a simple list of two values for either type of input. An integer value gets a .0 added, so it can be used in a Tcl expr and forced to be treated as a double. So an input like {1 2} will be assigned to the point_var as {1.0 2.0}.
  - **rect** is exactly four doubles. It can be in a single list or list of lists style, so {0 1 2 3}, {{0 1} {2 3}} or {{0 1 2 3}} are allowed. The rect_var will always be assigned as a simple list of four values.
- An **enum** type must include the list of legal enum values like this {**enum** val1 val2 ...}. It supports an optional list_type of * (a list of 0 or more) or **+** (a list of 1 or more) as the last enum value. It allows an optional {**hidden** val1 val2 ...} list for hidden enum values. The enum value inputs are case-insensitive, but the return value assigned to *var* is always forced to be all lower-case characters.
  For example,

  ```
  {sort_var name {-sort} {enum name age {hidden city state} +} optional {help description}}
  ```

  would allow `-sort age`, or `-sort "name CITY"`, but would give errors for `-sort ""`, or `-sort abc`.
  The help string will show the enum values but not the hidden enum values like this:

  ```
  -sort {name age} # help description (enum+, optional)
  ```

- An **object** type allows object names and pointers to be passed in as values like this {**object** *obj_type1 [obj_type2 ...]* [*list_type*]}. It supports an optional *list_type* of * (a list of 0 or more) or **+** (a list of 1 or more) as the last value. If *list_type* is not given, exactly one value is required for input. For example,

  ```
  {driver_var {} {-drivers} {object pin port *} optional {help description}}
  ```

  would allow a list of 0 or more pin or port objects for input. The input could be a list of pin or port pointers, or a name-pattern like "i1/p*" that allows * and ? pattern matching. *driver_var* will be assigned a list of pointers of the correct type (or types). So both of these uses would have the same result:

  ```
  -drivers [concat [get_db pins i1/p*] [get_db ports i1/p*]]
  ```

  ```
  -drivers i1/p*
  ```

  In both cases, all the pins and ports that match i1/p* would be returned as a list of objects and assigned to *driver_var*.

  If any objects are passed in that are not pin or port objects, an error will occur during parsing.

  If at least one pointer is required (e.g. list_type **+** or no list_type), and none are matched by the name pattern, an error will occur during parsing.

  Design objects like **hinst, inst, hnet, net, hpin, pin**, **port**, etc and library objects like **base_cell, base_pin, lib_cell, lib_pin**, etc, and technology objects like **layer**, **route_rule**, etc. are supported, but only objects that have a .name attribute can use a name-pattern in addition to pointers (e.g. a wire object has no .name so it cannot be looked up with a name-pattern). See the Stylus Common UI Database Reference for a list of all the obj_types.

- *flags* is list of flags that may be an empty list. The supported flags are
  - **optional** means the argument is optional. If the argument is not given, *var* will be set to the *default* value. Arguments are **optional** by default, so this is mostly to make it easy to see it in the definition. It cannot be combined with **required**.
  - **required** means the argument is required. It is an error to not have the argument.
  - **multiple** means the argument can appear multiple times and *var* will be set to a list of all the values.
  - **hidden** means the argument is not shown in the help output, partial argument name matching will not match it, and <tab> complete will not show it. **hidden** must be optional, so it cannot be combined with **required**, **required_group,** or **exclusive_required_group** flags.
  - **obsolete** means the argument still works but is considered obsolete. It will give a warning message if it is used that the user should stop using this argument to be compatible with future releases. An **obsolete** argument will automatically be **hidden** and cannot be a required argument. See **alias** if you just want to change the *arg_name* to a new *arg_name*.
  - {**exclusive_group** *group_name*} means only one of the arguments can be given from the same *group_name*. For example, if they are both optional, but only one of -my_int or -my_int2 is allowed:

    ```
    {int_var 0 {-my_int int} int {{exclusive_group
    group1}} {help for -my_int}}
    {int_var2 0 {-my_int2 int} int
    {{exclusive_group group1}} {help for -
    my_int2}}
    which would show up in the help usage as [-
    ```

```
my_int <int> | -my_int2 <int>]
```

- {**exclusive_required_group** *group_name*} means exactly one of the arguments must be given from the same *group_name.* For example, if exactly one of -my_int or -my_int2 is required:

  ```
  {int_var 0 {-my_int int} int
  {{exclusive_required_group group1}} {help for
  -my_int}}
  {int_var2 0 {-my_int2 int} int
  {{exclusive_required_group group1}} {help for
  -my_int2}}
  which would show up in the help usage as {-
  my_int <int> | -my_int2 <int>}
  ```

- {**required_group** *group_name*} means one or more of the arguments must be given from the same *group_name.*

- {**requires** *arg_name1* [*arg_name2* ...]} means this argument requires all the other arguments in the list. For example both -arg1 and -arg2 are required if -my_int is given.

  ```
  {int_var 0 {-my_int int} int {optional
  {requires -arg1 -arg2}} {help for -my_int}}
  ```

- {**alias** *obsolete_name [***warn***]*} means an alias of *obsolete_name* to the *arg_name* will be added. If **warn** is given, a warning message will be given if the *obsolete_name* is used telling the user to change to *arg_name,* otherwise *obsolete_name* will behave just like *arg_name*. The *obsolete_name* is not shown in help output. This can be used to migrate users to new argument names, but keep the old name for backward compatibility. For example, to migrate from `-old_my_int` to `-new_my_int` you would use this:

  ```
  {int_var 0 {-new_my_int int} int {{alias -
  old_my_int}} {help for -new_my_int}}
  ```

	• *help* is a help string for the argument, such as *"a file name to read"*.
-description *proc_help*	Specifies the help description for the defined procedure, shown by "help cmd" or "cmd -help". *Data_type*: string, optional
-extended_help *help_description*	Provides the extended help description for a procedure. It is printed after -description and arg_list; shown by help cmd or cmd -help. *Data_type*: string, optional
-help	Outputs the command usage and a brief description about the command parameters.
-hidden	Makes the procedure hidden and not public. That means <tab> complete will not return this proc name, and "info commands *" will not return this proc name. But if you type the full name like "info commands full_cmd_name" it will return the command, or "full_cmd_name -help" will return help for it. *Data_type*: bool, optional
proc_body	Defines the Tcl procedure body that uses var names from *arg_list*. *Data_type*: string, required
proc_name	Specifies the name of the Tcl procedure. *Data_type*: string, required

Example

Here is an example of some common usage:

```
define_proc test –description “An example of the extended argument syntax” {
   {out_file "default.out" out_file string optional "The output file name"}
   {my_count 0 {-count num_inputs} int required "The number of inputs"}
   {verbose false -verbose bool optional "Verbose output"}
```

```
  {check_err not_set {-check_err bool} bool optional "Override global check setting"}
  {effort low -effort {enum low medium high} optional "How much effort to use"}
  {my_insts {} {-insts} {object inst +} optional "One or more insts"}
}{
# Proc body goes here
  ...
}
```

The above procedure generates the following help output:

```
> test -help
```

An example of the extended argument syntax

```
Usage: test [-help] [<out_file>] [-check_err <bool>] -count <num_inputs> [-effort {low
medium high}] [-verbose]

-help                       # Prints out the command usage
<out_file>                  # The output file name (string, optional)
-check_err <bool>           # Override global check setting (bool, optional)
-count <num_inputs>         # The number of inputs (int, required)
-effort {low medium high}   # How much effort to use (enum, optional)
-insts <inst>+              # One or more insts (Iinst)+, optional)
-verbose                    # Verbose output (bool, optional)
```

Related Information

- define_proc_arguments
- parse_proc_arguments

define_proc_arguments

```
define_proc_arguments
command_name
[-info string]
[-define_args string]
[-hide]
```

Defines the arguments and attributes of a Tcl procedure. The arguments and attributes are displayed when you view the help associated with the procedure.

The `define_proc_arguments` command, when used with the `parse_proc_arguments` command, enables you to validate command usage. For invalid commands, appropriate error messages are displayed.

Parameters

proc_name	Specifies the name of an existing Tcl procedure. *Data_type*: string, required
-info *string*	Provides help information for the procedure. *Data_type*: string, optional
-define_args *string*	Defines all possible arguments of the Tcl procedure. It can also be used to define the help text and constraints for individual arguments of the procedure and generates the help for command options. The variable *arg_defs* is a list of lists where each list element defines one argument. Each element in the list has the following format: `"arg_name" "option_help" "value_help" "data_type" "attributes"` *Data_type*: string, optional
	`arg_name`: Specifies the name of the argument.
	`option_help`: Provides a short description of the argument.
	`value_help`: Specifies the help information for the associated option values.

	data_type: (Optional) Used for validation of options. Supported data types are `string`, `list`, `boolean`, `int`, `integer`, `float`, or `one_of_string`. *Default:* `string`
	attributes: (Optional) Used for validation of options. The attributes can be: • `required`: This argument must be specified . This is mutually exclusive with `optional` and `internal`. • `optional`: Specifying this argument is optional. This is mutually exclusive with `required`. • `internal`: Disables the display of help information for the option in the procedure help. This argument is mutually exclusive with `required`. • `value_help`: Indicates that the valid values for `one_of_string` argument should be listed whenever argument help is shown. • `values {`*`allowable_values`*`}`: This argument must be used with the `one_of_string` data type. • `merge_duplicates` : Ensures that when the option appears more than once in a command, its values are concatenated into a list of values. The default behavior is that the right-most value specified for the option is used. • `bind_option` *`other_arg_name`*: Binds the current option with the option specified as the *`other_arg_name`* variable. • `mutual_exclusive_group` *`group_name`*: Defines a group name and includes the current option in the *`group_name`* group. All options in the same group are treated as mutually exclusive. • *Default:* `optional`
`-hide`	Hides the procedure and its body. *Data_type*: bool, optional

Examples

The following Tcl procedure uses the `define_proc_arguments` command for defining arguments and attributes, and for validating the command usage.

```
proc cmdProc {args} {

parse_proc_arguments -args $args results

foreach argname [array names results] {

puts "$argname = $results($argname)"

}

}

define_proc_arguments cmdProc -info "Command Description" \

-define_args \

{{-string "String help" "string_val" string optional}

{-integer "Int help" "int_value" int optional}

{-list "List help" "list_value" list {optional bind_option -boolean_a}}

{-boolean_a "BoolA help" "" boolean {optional mutual_exclusive_group

g0}}

{-boolean_b "BoolB help" "" boolean {optional mutual_exclusive_group

g0}}

{-float_num "Float help" "float_value" float}

{-oosa "One of StringA help" "one_of_str_value" one_of_string

{optional value_help {values {v1 v2}}}}

{-oosb "One of StringB help" "one_of_str_value" one_of_string

{optional {values {v4 v5}}}}

}
```

Now, if the `cmdProc` procedure is invoked with the `-help` option at the command prompt, the following help information is displayed.

```
cmdProc -help

Usage: cmdProc # Command Description
```

```
[-string string_val] (String help)

[-integer int_value] (Int help)

[-list list_value] (List help, Bind_Option=-Bool)

[-boolean_a ] (BoolA help, Mutual_Exclusive_Group=g0)

[-boolean_b ] (BoolB help, Mutual_Exclusive_Group=g0)

[-float_num float_value] (Float help)

[-oosa one_of_str_value] (One of StringA help, Values={v1 v2 })

[-oosb one_of_str_value] (One of StringB help)
```

Related Information

- define_proc
- parse_proc_arguments

define_variables

```
define_variables
[-help]
name_list
[-transient  | -delete ]
```

Creates Tcl variables in the global namespace if they do not already exist, and adds them to a save list to be saved by `write_db` and restored by `read_db`. The variables will be saved even if the root attribute `write_db_auto_save_user_globals` is false. It can also remove Tcl globals that would be automatically saved if `write_db_auto_save_user_globals` is true. It is intended for Tcl global variables, but it also supports namespace variables.

During `read_db`, the Tcl variables are created if they do not already exist and the value is set. If a namespace variable is given, it will also create the namespace if it does not already exist.

Parameters

`-delete`	Deletes the Tcl variables and removes them from the save and transient lists. It returns an error if the Tcl variables do not exist. *Data_type*: bool, optional
`-help`	Outputs a brief description that includes type and default information for each `define_variables` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man define_variables`
`name_list`	Specifies a list of Tcl variable names. If the specified Tcl variable does not already exist, it is created and added to a save list of variables to be saved by `write_db`. *Data_type*: string, required
`-transient`	Adds the Tcl variables to a transient list used by `write_db` (and removes them from the save list if they are there). The transient list will override the automatic save list created if the root attribute `write_db_auto_save_user_globals` is `true`, so the variables will not be saved. *Data_type*: bool, optional

Examples

The following command saves the variables `global1` and `namespace1::var1` with `write_db`:

```
>define_variables {global1 namespace1::var1}
```

Related Topics:

- read_db
- `write_db`

delete_floating_constants

```
delete_floating_constants
-help
```

Deletes dangling 1'b1s or 1'b0s from the Verilog. They are either dangling nets assigned 1'b1 or 1'b0 or 1'b1/1'b0 connections that are not hierarchically connected to a leaf instance.

Parameters

None

Example

In the code below only BUFX2 is a leaf.

```
//
// Original Verilog
module hier4 (A,B);
input A;
input B;

BUFX2 u0 (.A(A));
BUFX2 u1 (.A(1'b1));
endmodule

module hier3 ();

hier4 u0 (.A(1'b1),.B(1'b0));
hier4 u1 (.A(1'b0),.B(1'b1));
endmodule

module hier2 ();

hier3 u0 ();
endmodule

module top ();

// Internal wires
wire a;
wire b;
wire c;
wire d;
```

```
assign a = 1'b1 ;
assign b = 1'b1 ;
assign c = 1'b0 ;
assign d = 1'b0 ;

hier2 u0 ();
BUFX2 u1 (.A(a));
BUFX2 u2 (.A(c));
endmodule

//
// Verilog after delete_floating_constants is issued

module hier4 ( A, B);
input A;
input B;

BUFX2 u0 (.A(A));
BUFX2 u1 (.A(1'b1));
endmodule

module hier3 ();

hier4 u0 (.A(1'b1));
hier4 u1 (.A(1'b0));
endmodule

module hier2 ();

hier3 u0 ();
endmodule

module top ();

// Internal wires
wire a;
wire c;

assign a = 1'b1 ;
assign c = 1'b0 ;

hier2 u0 ();
BUFX2 u1 (.A(a));
BUFX2 u2 (.A(c));
endmodule
```

delete_dangling_ports

```
delete_dangling_ports
-help
-partition
```

Deletes ports that are disconnected from both the inside and outside of a module, and ports that are connected within the module but disconnected outside the module.

Parameters

`-help`	Prints a brief description that includes type and default information for each `delete_dangling_ports` parameter. For a detailed description of the command and all of its parameters, use the man command: `man delete_dangling_ports`
`-partition`	Specifies the partition only flag of the type boolean. *Data_type*: bool, optional

Related Information

- `delete_empty_hinsts`
- `disconnect_dangling_port`

deselect_obj

```
deselect_obj
[-help]
{[-all ] | [obj_list]}
```

Deselects the object or the list of objects from a set. The object type can be any object type drawn on GUI such as instance wire, via instance...

Parameters

`-help`	Prints a brief description that includes type and default information for each `deselect_obj` parameter. For a detailed description of the command and all of its parameters, use the man command: `man deselect_obj`
`-all`	Deselects all selected objects. *Data_type*: bool, optional
obj_list	Specifies the list of objects to be deselected. *Data_type*: (any_object)+, optional

Example

The following command deselects the instance name i1.

```
deselect_obj [get_db insts i1]
```

Related Information

- `delete_obj`
- `report_obj`
- `select_obj`

encrypt

```
encrypt
[-help]
input_file_name
[-pragma]
[> file]
[-tcl  | -verilog ]
```

Encrypts a file in an nc-protect compatible format.

Parameters

`-help`	Prints out the command usage.
input_file_name	Specifies the file to be encrypted. *Data_type*: string, required
> file	Specifies the name of the encrypted file. By default, the encrypted file is printed to standard out (stdout). *Data_type*: redirect_operator, optional
`-pragma`	Specifies to only encrypt the text between the protect begin and protect end NC Protect pragmas. Encryption block must start with pragma protect comment. *Data_type*: bool, optional
`-tcl`	Specifies that the file to be encrypted is a Tcl file. This is the default option. *Data_type*: bool, optional
`-verilog`	Uses Verilog style comments for NC Protect pragmas. *Data_type*: bool, optional

Related Topics

- decrypt

eval_legacy

```
eval_legacy
[-help]
script
```

Runs a legacy command in the legacy Tcl interpreter. This command is not required in normal usage. However, it may be needed in cases where some legacy usage is not yet available in the Stylus Common UI. This command is intended as a temporary workaround as users transition to the Stylus Common UI and will be removed in a future release.

You should first check if there is a Stylus Common UI equivalent command before using this command by using the `get_common_ui_map` command:

```
>get_common_ui_map <legacyCommand>
```

Most legacy commands are available, but legacy commands that have behavior changes in the Stylus Common UI will not work properly (`saveDesign`, `restoreDesign`, `init_design`, `report_timing` etc.).

The legacy commands run in a separate Tcl interpreter, so Tcl variables in the current interpreter are not available. That means the *script* should normally be inside " " rather than { } so Tcl variable substitution occurs in the current interpreter before passing the script to the legacy Tcl interpreter.

Note: You can use `1` as an abbreviation for the `eval_legacy` command. To use `eval_legacy` in Stylus, you can simply write `1` and then the legacy command name. It also enables the command line tab completion for legacy commands after `eval_legacy`. The following type of input is supported:

```
> l{legacyCmd<tab> -option<tab>}
```

You can also use `eval_legacy {restoreDesign/ saveDesign}` to run legacy `restoreDesign` and `saveDesign` commands.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `eval_legacy` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man eval_legacy`
`script`	Specifies one or more legacy Tcl commands to run in the legacy Tcl interpreter. *Data_type*: string, required

Example

- The following command run `setFillerMode` and `addFiller` commands in the legacy Tcl interpreter.

  ```
  set my_prefix fill eval_legacy "setFillerMode -corePrefix $my_prefix"
  eval_legacy "addFiller ..."
  ```

  You can have more than one legacy command inside a single eval_legacy like this:

  ```
  set my_prefix fill
  eval_legacy
           "setFillerMode -corePrefix $my_prefix
            addFiller ..."
  ```

- The following uses a TCL variable defined in the current interpreter and passes it to a legacy command

  ```
  set inst CTS_ccl_BUF_clk_G0_L2_245
  eval_legacy “dbGet top.insts.name $inst”
  ```

- The following uses `get_db` to find a particular instance and then passes that object's name to legacy `dbGet`. The argument is inside “”, so both of the `[get_db.. ]` proc substitutions, and the `$inst` are evaluated inside the CUI interpreter first, before passing the result to the `dbGet` command in the legacy interpreter.

  ```
  eval_legacy "dbGet -p1 top.insts.name  [get_db [get_db insts $inst] .name]"
  ```

Related Information

- get_common_ui_map
- convert_legacy_to_common_ui

get_all_layers

```
get_all_layers
[-help]
[{object display multiple_color internal module metal color custom oa_layout}]
```

Returns a complete list of all layers and floorplan object settings. If you specify type, the software returns all the layers of the specified type. This command can be used at any stage in the design flow.

Parameters

`-help`	Prints a brief description that includes type and default information for each `get_all_layers` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man get_all_layers`
`{object display multiple_color internal module metal color custom oa_layout}`	Specifies the type of the layer. Innovus supports nine types of layers, which can be specified as follows: • `object`: Returns object layers, which represent db objects, such as instances, modules, pins, and so on. • `display`: Returns display-only or view-only layers, including flightlines, rulers, and congestion. • `multiple_color`: Returns multiple color layers, such as congestion and yield maps. • `internal`: Returns internal layers, such as bump layers, structured data path layers, etc. • `module`: Returns module layers, which are layers associated with hinsts. • `metal`: Returns wire/via layers, including metal/via, pin, and blockage. • `color`: Returns color layers. • `custom`: Returns custom layers, which are used to represent custom objects and GDSII data. • `oa_layout`: Returns OA layout layers. *Data_type*: enum, optional

Example

Returns all metal layer names:

```
get_all_layers metal
```

get_common_ui_map

```
get_common_ui_map
-help
legacy_name
```

Provides the mapping of a legacy command and its options or the dbget chain in Stylus Common UI.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `get_common_ui_map` parameter. For a detailed description of the command and all of its parameters, use the man command: `man get_common_ui_map`
`legacy_name`	Specifies the legacy command name, the UDM name, or the dbget chain. *Data_type*: string, required

Examples

- The following command gets the mapping of the `assembleDesign` command in Stylus Common UI:

  ```
  >get_common_ui_map assembleDesign
  ```

  Output:

  ```
  assembleDesign                    assemble_design
  -allTimingBlocks                  -all_timing_blocks
  -block                            -oa_block
  -blockCell                        -block_cell
  -blockData                        -block_data
  -blockDir                         -block_dir
  -cpfFile                          -cpf_file
  -exceptBlocks                     -ignore_timing_blocks
  ```

-fe	-encounter_format
-fplan	-floorplan_only
-keepPGPinGeometry	-keep_pg_pin_geometry
-keepPinGeometry	-keep_pin_geometry
-mmmcFile	-mmmc_file
-noDefMerge	-no_def_merge
-skipCells	-skip_cells
-skipComponents	-skip_components
-skipNets	-skip_nets
-skipSpecialNets	-skip_special_nets
-topDesign	-top_design
-topDir	-top_dir

- The following command gets the mapping of the dbget chain (`dbget top.nets.vWires.objType`) in Stylus Common UI:

```
> get_common_ui_map "dbGet top.nets.vWires.objType"                    #
support dbGet and strict legacy db attribute name
Output:
dbGet top.nets.vWires.objType   get_db nets .virtual_wires.obj_type
```

Related Information

- `get_ccopt_property_common_ui_map`

get_compress_level

`get_compress_level`

Displays the current compression speed level used for compressing large data files. Compression speed is controlled with the `set_compress_level` command.

Related Information

- `set_compress_level`

get_layer_preference

```
get_layer_preference
[-help]
layer_name
{-is_visible | -is_selectable | -color | -line_width | -stipple | -stipple_data}
```

Provides information about a single preference for a single object.

See `set_layer_preference` for descriptions of how to set layer preferences.

Parameters

`-help`	Prints a brief description that includes type and default information for each `get_layer_preference` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man get_layer_preference`
`-color`	Provides information about the color of the object. A color name or a numeric RGB value is returned. *Data_type*: bool, optional
`-is_selectable`	Provides information about whether the object is selectable. If a value of `1` is returned, the object is selectable. If a value of `0` is returned, the object is not selectable. *Data_type*: bool, optional
`-is_visible`	Provides information about whether the object is visible. If a value of `1` is returned, the object is visible. If a value of `0` is returned, the object is not visible. *Data_type*: bool, optional
`layer_name`	Provides the name of the layer or the object type. *Data_type*: string, required
`-line_width`	Provides information about the number of pixels in the border of the specified object. An integer value is returned. *Data_type*: bool, optional
`-stipple`	Provides information about the stipple pattern of the object. A stipple name is returned. *Data_type*: bool, optional
`-stipple_data`	Provides information about the width and height of the stipple pattern. This information is in x-window bitmap format. *Data_type*: bool, optional

Example

Returns the color for `M1` objects:

```
get_layer_preference M1 -color
```

Related Information

- `set_layer_preference`

get_legacy

```
get_legacy
-help
common_ui_cmd_attr_name
```

Returns the legacy command, legacy Tcl global, or legacy `setXXXMode -option` that corresponds to the specified Stylus Common UI command or root attribute.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `get_legacy_command` parameter. For a detailed description of the command and all of its parameters, use the man command: `man get_legacy`
`common_ui_cmd_attr_name`	Specifies the Stylus Common UI command name or attribute name. *Data_type*: string, required

Examples

- The following command returns the legacy command that corresponds to the Stylus Common UI command `place_design`:

  ```
  @innovus 1> get_legacy place_design
  Command: place_design
  ```

- The following command returns the `setFillerMode` option that corresponds to the Stylus Common UI attribute `add_fillers_prefix`:

  ```
  @innovus 2> get_legacy add_fillers_prefix
  Mode option: setFillerMode -corePrefix
  ```

- The following command returns the legacy Tcl global that corresponds to the Stylus Common UI attribute `init_lef_files`:

  ```
  @innovus 3> get_legacy init_lef_files
  Tcl global: init_lef_file
  ```

get_message

```
get_message
-help
-id msgID
[-severity | -limit | -suppress | -count | -short | -long | -on_error  | -
stop_script_limit | -detail ]
```

Gets the limit, severity, or suppression for a message.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `get_message` parameter. For a detailed description of the command and all of its parameters, use the man command: `man get_message`
`-count`	Returns the number of times this message has been issued this session.
`-detail`	Returns the message of specified id.
`-id` *msgID*	Specifies a single message ID, in the form "ENCFOO-123".
`-limit`	Returns the limit for the specified message as an integer. -1 is unlimited. 0 is suppressed.
`-long`	Returns the long message description, as used for the Description section in "`man` *msgID*"
`-on_error`	Returns the error action for the specified message.
`-severity`	Returns the severity, `error`, `warn`, `info`, or `default`, of the specified message as a string.
`-short`	Returns the short message description. Equivalent to "`help` *msgID*" .
`-stop_script_limit`	Returns the message limit allowed for every public command. Message limit set as -1 means unlimited.
`-suppress`	Returns `1` if the message is suppressed, `0` if not suppressed.

Related Information

- `set_message`

get_preference

```
get_preference
[-help]
preference_name
```

Returns the current preference settings.

Parameters

`-help`	Prints out the command usage
preference_name	Specifies the name of a preference. *Data_type*: string, required

Related Information

- set_preference

gui_bind_key

```
gui_bind_key
[-help]
key
-cmd cmdOrProcName
[-is_public]
```

Enables you to create keyboard shortcuts.

You can use this command at any time after starting Innovus.

Parameters

`-help`	Prints a brief description that includes type and default information for each `gui_bind_key` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man gui_bind_key`

key	Specifies the key to be used as the shortcut. You can specify a: • Single key-Specify any one of the following keys: ◦ `a-z` ◦ Function keys: Use `F1-F12` ◦ Arrow keys: Use `Up Down Left Right` ◦ Number keys ◦ Unshifted punctuation keys: `. , / - =` ◦ Other keys: Use `Space`, `Delete`, `Backspace`, or `Esc` keys • Combination-Specify `Shift` as the first key followed by '-' followed by: ◦ `A - Z` ◦ One of the 'shifted' punctuation chars: `~@#$%^&*()_+` ◦ Function keys: use `F1 - F12` ◦ Arrow keys: use `Up Down Left Right` ◦ Other keys: use `Space`, `Delete`, `Backspace`, or `Esc` • Ctrl/Alt Key Combination-Specify `Ctrl` or `Alt` as the first key followed by '+' followed by any of the single keys above *Data_type*: string, optional
`-cmd` *cmdOrProcName*	Specifies the command or proc to be executed - no parameters allowed. *Data_type*: string, optional
-is_public	Specifies the bindkey as user defined. *Data_type*: string, optional

Example

Runs the `gui_fit` command when you press `f` on your keyboard:

```
gui_bind_key f "gui_fit"
```

gui_bind_key

gui_get_record_obj_by_info

```
gui_get_record_obj_by_info
[-help]
[-layer layer_value]
[-name object_name]
-obj_type {viainst sviainst swire wire pwire pblkg}
-rect {x1 y1 x2 y2}
```

Returns the point of an object based on specific layer/name/area.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `gui_get_record_obj_by_info` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man gui_get_record_obj_by_info`.
`-layer layer_value`	Specifies the layer value or layer range. *Data_type*: string, optional
`-name object_name`	Specifies the object name of net name for wire/swire/pwire/viainst/sviainst. *Data_type*: string, optional
`-obj_type {viainst sviainst swire wire pwire pblkg}`	Specifies the object type. *Data_type*: enum, required
`-rect {x1 y1 x2 y2}`	Specifies the coordinates of the object bounding-box. *Data_type*: rect, required

Example

The following command returns the point of wire which area is `{365.83 989.66 384.59 989.94}`, layer value is `3`, net name is `DTMF_INST/TDSP_CORE_INST/MPY_32_INST/ab_b[6]`.

```
uiGetRecordObjByInfo -objType wire -rect {365.83 989.66 384.59 989.94} -layer 3 -name
{DTMF_INST/TDSP_CORE_INST/MPY_32_INST/ab_b[6]}
```

gui_move_cursor

```
gui_move_cursor
[-help]
x y
```

Places the cursor at the specified location in the main window.

You can specify this command at any time after starting the Innovus software.

Parameters

`-help`	Prints a brief description that includes type and default information for each `gui_move_cursor` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man gui_move_cursor`
x	Specifies the x coordinate of the cursor location. *Data_type*: float, required
y	Specifies the y coordinate of the cursor location. *Data_type*: float, required

Example

The following command places the cursor at the coordinates (`133`, `277`).

```
gui_move_cursor 133 277
```

help

```
help
[-help]
pattern1
[pattern2]
[-category category]
[-detail  | -summary ]
[-command  | -attribute  | -obj  | -message ]
```

Displays help information related to a command, object, attribute or message based on the specified arguments.

By default, only commands, objects and attributes are searched for help.

Parameters

`-attribute`	Searches attributes only. *Data_type*: bool, optional
`-category` *category*	Returns only those attributes that are in the given category. *Data_type*: string, optional
`-command`	Searches commands only. *Data_type*: bool, optional
`-detail`	Provides detailed help, even when multiple help strings are returned. By default, a short help description is returned if more than one items are matched. *Data_type*: bool, optional
`-message`	Searches messages only. Note, By default messages are not searched. This option is required to see the message help. *Data_type*: bool, optional
`-obj`	Searches objects only. *Data_type*: bool, optional
pattern1	Searches for any command, object, or attribute that matches pattern1. Pattern matching with * and ? is allowed. If the pattern ends in : (e.g. inst:) only objects will be matched. If the pattern starts with . (e.g. .place*) then only attributes will be matched. *Data_type*: string, required
pattern2	Returns help only for those attributes that match pattern2 for the objects matching pattern1. For example, the following command will return all attributes that match `place*` on the `inst` object. `help inst place*` *Data_type*: string, optional
`-summary`	Provides a short description even for a single match. By default, if only a single item is matched, the full help description is returned. *Data_type*: bool, optional

Examples

- The following command prints the help for any command, object, or attribute that matches "port*". For attributes, the owning object is shown in (), followed by a summary of the data_type, read/write status, default value, allowed `get_db -index` values, and a description:

```
innovus> help port*
Objects:
port: External logical ports of the design. See pg_ports for power/ground ports

Attributes:
port(bump) # port, read-only, default={}, indices={}
# The port the bump is assigned to.
ports(design) # port*, read-only, default={}, indices={}
# ports of design. Does not include pg_ports, unless PG port is explicitly in the
# Verilog netlist.
ports(root) # port*, read-only, default={}, indices={}
# Short-cut for [get_db current_design .ports]
```

- The following command shows the help output for the attribute `.base_cell` on the object `inst:`. The : and . chars in `inst:` and `.base_cell` are optional as they restrict the search to objects and attributes, respectively.

```
innovus> help inst: .base_cell
Attribute: base_cell (on obj_type: inst)
Description: The base_cell of the inst.
category:
default:
is_computed: false
is_settable: true
is_user_defined: false
skip_in_db: true
type: base_cell
```

- The following output shows the help for a single command:

```
innovus 14> help write_metric
Command: write_metric:

Usage: write_metric [-help] [<pattern>] [-format <string>] [-id <string>] [-merge]
[-names <string>] {-file <string> | -inline }

-help      # Prints out the command usage
<pattern>   # List of patterns to match metrics (string, optional)
-file <string> # Filename to write (string, optional)
-format <string> # Data format for the return value (string, optional)
```

```
-id <string> # Database ID to output (string, optional)
-inline # Return the result instead of writing it to a file (bool, optional)
-merge # Merge the results into an existing file (bool, optional)
-names <string> # List of patterns to match snapshot names (string, optional)
```

- The following command shows the help output for a message.

```
innovus 14> help -message IMPTCM-48
Error/Warning message: IMPTCM-48: "%s" is not a legal option for command "%s".
Either the current option or an option prior to it is not specified correctly.
```

is_common_ui_mode

```
is_common_ui_mode
[-help]
```

Checks if the Stylus Common UI mode is enabled. This command is helpful if you are using both Common UI and Legacy UI, and you want to source their respective script based on the value returned by `is_common_ui_mode`. In such scripts. if `is_common_ui_mode` returns 1, Stylus Common UI commands will be sourced, and if it returns `0`, Legacy commands will be sourced.

For example, you can write a script, which can be sourced irrespective of UI, as below.

```
if {[is_common_ui_mode]}

{
<common UI commands>
}

else

{
<Legacy UI commands>
}
```

Parameters

None

Example

The following example checks it on the Innovus prompt:

```
@innovus> is_common_ui_mode

1

innovus> is_common_ui_mode

0
```

Return value `1` indicates that the Stylus Common UI mode is enabled.

lminus

```
lminus
[-exact]
list1
list2
```

Subtracts the specified elements in the second list (*list2*) from the first specified list (*list1*) .

Parameters

-exact	Searches for an exact match.
list1	Specifies the list from which elements must be removed.
list2	Specifies the list of elements to be subtracted.

Examples

- The following command subtracts the elements from list l2 from list l1.

```
innovus> set l1 {a a* ab2 b1 b2 c}
innovus> set l2 {a c}
innovus> lminus $l1 $l2
a* ab2 b1 b2
```

- The following command subtracts a list consisting of a* and c from the list of elements of l1.

```
innovus> lminus $l1 {a* c}
b1 b2
```

- The following command uses an exact search.

```
innovus> lminus -exact $l1 {a* c}
a ab2 b1 b2
```

man

```
man
command_name | msg_id
```

This command is a Tcl wrapper that calls the Linux ‘man’ command, so you can see man pages for Innovus commands, Innovus root attributes, or other Linux commands. The man-page directories inside the Innovus installation are automatically added to the beginning of the Linux environment variable MANPATH so you will see Innovus man-pages before any other man-pages in your MANPATH.

The only exception is if the argument matches the msg_id format (e.g. several letters with a dash followed by an integer like ‘ABC-123’)., the Tcl wrapper directly returns the msg_id message you would see in the log file, along with any extended help description if it is available for that msg_id. These msg_id ‘man-pages’ are not included in the man-page directories, but are only available inside Innovus.

For all other arguments, it just calls the Linux ‘man’ command with the proper MANPATH, so you can use any Linux ‘man’ options as described in the Linux man-page for ‘man’.

For information on setting up MANPATH for Stylus Common UI commands directly from Linux, see Setting Up the Run-Time Environment in the *Getting Started* chapter of the *Innovus Stylus Common UI User Guide*.

Parameter

command_name	Indicates the name of the text command (or root attributes) for which the manual information is required.
msg_id	Specifies a message ID in the format like ‘ABC-123’. The output will show the log file message and any extended help description that can include likely causes, possible fixes, and possible workarounds for this msg_id.

Example

- The following command returns the manual description of the command `report_timing`:

  ```
  man report_timing
  ```

- The following command returns the manual description of the `SOCSYC-139` message ID:

  ```
  man SOCSYC-139
  ```

Related Information

- Setting Up the Run-Time Environment in the *Getting Started* chapter of the *Innovus Stylus Common UI User Guide*.

parse_proc_arguments

```
parse_proc_arguments
[-help]
result_array
[forced_proc]
-args string
```

Supports argument validation within a Tcl procedure and enables the use of the `-help` option to view the help information associated with the procedure.

Procedures using the `parse_proc_arguments` command will validate the arguments. If the required arguments are missing or have invalid values, the `parse_proc_arguments` command generates a Tcl error and the calling procedure terminates.

The `parse_proc_arguments` command must be the first command called in the procedure. This command cannot be used outside of a procedure.

Parameters

`-help`	Prints out the command usage.
`-args string`	Specifies the list of arguments passed to the Tcl procedure. *Data_type*: string, required
result_array	Specifies the result array into which the parsed arguments are to be stored. *Data_type*: string, required
forced_proc	Specifies the name of the invoking procedure. *Data_type*: string, optional

Examples

The following Tcl procedure uses the `parse_proc_arguments` command for argument validation.

```
proc cmdProc {args} {

parse_proc_arguments -args $args results

foreach argname [array names results] {
```

```
puts "$argname = $results($argname)"

}

}


define_proc_arguments cmdProc -info "Command Description" \

-define_args \

{{-string "String help" "string_val" string optional}

{-integer "Int help" "int_value" int optional}

{-list "List help" "list_value" list {optional bind_option -Bool}}

{-boolean_a "BoolA help" "" boolean {optional mutual_exclusive_group g0}}

{-boolean_b "BoolB help" "" boolean {optional mutual_exclusive_group g0}}

{-float_num "Float help" "float_value" float}

{-oosa "One of StringA help" "one_of_str_value" one_of_string {optional value_help
{values {v1 v2}}}}

{-oosb "One of StringB help" "one_of_str_value" one_of_string {optional {values {v4
v5}}}}

}
```

Now, if the cmdProc procedure is invoked using the following command at the command prompt, argument validation takes place and the following errors are reported.

```
innovus> cmdProc -boolean_a -boolean_b

**ERROR: Options '-boolean_a -boolean_b' are mutually exclusive for 'cmdProc'.

Usage: cmdProc [-string string_val] [-integer int_value] [-list list_value] [-boolean_a
]

[-boolean_b ] [-float_num float_value] [-oosa one_of_str_value {v1 v2 }] [-oosb
one_of_str_value]]

**ERROR: Errors in 'cmdProc'.
```

The following command to invoke the cmdProc procedure will also return the errors shown below.

```
innovus> cmdProc -oosa v5

**ERROR:  The arguments specified (v5) in cmdProc -oosa are not valid. Refer to the
documentation for valid arguments.
```

```
Usage: cmdProc [-string string_val] [-integer int_value] [-list list_value] [-boolean_a
]

[-boolean_b ] [-float_num float_value] [-oosa one_of_str_value {v1 v2 }] [-oosb
one_of_str_value]]

**ERROR: Errors in 'cmdProc'.
```

print_message

```
print_message
-help
formatString
[args]
[-error]
[-warning]
```

Generates a message that is included in the `report_messages` output if the message is designated as an error or a warning.

Parameters

args	Specifies the format argument value to print. *Data_type*: string, optional
-error	Specifies the message is an error. A prefix will be added. *Data_type*: bool, optional
formatString	Specify the format specifier like sprintf or Tcl's format command. *Data_type*: string, required
-help	Prints a brief description that includes the type and default information for each `print_message` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man print_message`.
-warning	Specifies that the message is a warning. A prefix will be added. *Data_type*: bool, optional

Examples

The following commands generate an error and a message, which are included in the report_message output:

```
[DEV]innovus 3> print_message -error "A test %s." "from the user"
ERROR: (user): A test from the user.


[DEV]innovus 4> print_message -warning "A test %s." "from the user"
WARN: (user): A test from the user.


[DEV]innovus 5> print_message "A test %s." "from the user"
A test from the user.


[DEV]innovus 6> report_messages
*** Summary of all messages that are not suppressed in this session:
Severity      ID       Count          Summary
ERROR       no_id       1            A test %s.
ERROR       user        1            A test %s.
WARN        user        1            A test %s.
*** Message Summary: 1 warning(s), 2 error(s)
```

Related Information

- report_messages

program Category Attributes

The root attributes in the program category are:

- cmd_file
- log_file
- logv_file
- program_major_version
- program_name
- program_short_name
- program_version
- selected

To set the value of any root attribute, use the `set_db` command:

```
set_db attribute_name value
```

To get the current value of any root attribute, use the `get_db` command:

```
get_db attribute_name
```

To get the current value of all attributes in the `program` category, use the following command:

```
get_db -category program
```

cmd_file

`cmd_file` *fileName*

Default: ""

Data_type: string, read/write

Specifies the cmd log file name of the program.

The `.cmd` file captures just the Tcl commands executed without extra formatting so they can be cut-and-pasted for reuse. The names of the files default to the program name, extension, and extra number if there is a file name collision (e.g. innovus.log2, innovus.logv2, innovus.cmd2). These names can be overridden, or the files suppressed with the Linux command line options `-log`, `-no_cmd`, and `-no_logv` (see 'man innovus' for details).

The `cmd_file` root attribute is writable. You can change where the output is logged for any of them

during the run.

log_file

`log_file` *`fileName`*

Default: ""

Data_type: string, read/write

Specifies the log file name of the program. It will default to `innovus.log<N>` where `<N>` is a unique integer to avoid file name collisions. It can be set from the Innovus Linux command line, and it can be changed in Tcl during the run to direct the log output to different files. If it is changed, the new file is opened for write (not append), so any previous contents are lost.

The `log_file` root attribute is writable. You can change where the output is logged for any of them during the run.

logv_file

`logv_file` *`fileName`*

Default: ""

Data_type: string, read/write

Specifies the verbose log file name of the program.

The `.logv` file captures all output to the xterm and has more detailed verbose output that is not normally needed. However, it can be useful when debugging problems. Every output line has a time-stamp, and complex commands like `opt_design` will write out much more detailed internal information to help debug issues.

The `logv_file` root attribute is writable. You can change where the output is logged for any of them during the run.

program_major_version

`program_major_version`

Default: ""

Data_type: double, read-only

Returns the major version and one digit after the decimal point.

For example, the following command gives the major version of the software:

```
@innovus 1> get_db program_major_version
20.1
```

program_name

```
program_name
```

Default: ""

Data_type: string, read-only

Specifies the name of the program.

For example, the following command gives the name of the software:

```
@[DEV]innovus> get_db program_name
Innovus Implementation System
```

program_short_name

```
program_short_name
```

Default: ""

Data_type: string, read-only

Specifies the short name of the program.

For example, the following command gives the short name of the software:

```
@[DEV]innovus> get_db program_short_name
innovus
```

program_version

```
program_version
```

Default: ""

Data_type: string, read-only

Specifies the version of the program.

For example, the following command returns the software version:

```
@innovus 1> get_db program_version
20.10-d255_1
```

selected

`selected` *`list`*

Default: ""

Data_type: string, read/write

Specifies the list of currently selected objects. It can be set to a list of objects as long as they are selectable.

puts

`puts`
`[-help]`
`[-nonewline]`
`[-logv]`
`[`*`channel`*`]`
`string`

Outputs information to the screen (stdout) and Innovus log files (.log/.logv).

Parameters

`-help`	Prints a brief description of the command. For a detailed description of the command, use the `puts` command: `man puts`
`channel`	Specifies a channel id returned by `open` or `socket`. *Data_type*: string, optional
`-logv`	Specifies to write string to the .logv file only. This option cannot be used with *`channel`*. *Data_type*: bool, optional
`-nonewline`	Suppresses the new-line that is normally added after the *`string`* value. *Data_type*: bool, optional
`string`	Specifies a string value to write out. *Data_type*: string, optional

redirect

```
redirect
[-help]
[file_or_var_name]
[command]
[-tee]
[-stdin | -append]
[-stdin | -variable]
[-variable]
[-stderr | -stdin]
```

Redirects the output to a file or variable. You can also directly use redirect characters without explicit use of the redirect command for Innovus commands.

Any Tcl return value from *command* is returned by redirect. If redirect is used at the prompt, any echoing of the Tcl return at the prompt is not redirected.

You can also use the `redirect` command to write the contents directly to a gzip compressed file by ending the file name with .gz.

Parameters

`-help`	Prints a brief description that includes type and default information for each `redirect` parameter.
`file_or_var_name`	File or variable name for output to go. *Data_type*: string, optional
`command`	Command to be executed. *Data_type*: string, optional
`-append`	Appends stdout/stderr to file or variable. *Data_type*: bool, optional
`-stderr`	Redirects stderr. *Data_type*: bool, optional
`-stdin`	Redirects stdin. *Data_type*: bool, optional
`-tee`	Redirects output to both screen and file or variable. *Data_type*: bool, optional
`-variable`	Redirects to a Tcl variable. *Data_type*: bool, optional

It is also possible to use redirect characters similar to Unix usage, for Innovus commands, but not for built-in Tcl commands. This usage works for any Innovus command that does not explicitly register > or >> as options (if > or >> are visible in the man page or -help output for the command, it is explicitly registered). Note that the "source" command is overlaid by Innovus to add various verbose options, so it also supports these characters.

The table below shows the usage of redirect characters, and how to use & to indicate a Tcl var name.

Redirect Characters	Meaning
> <name>	redirect stdout to the file <name>

>> <name>	redirect stdout and append to the file <name>
> &<name>	redirect stdout to the Tcl var <name>
>> &<name>	redirect stdout and append to the Tcl var <name>

Examples

- Redirects the output to a new file timing.txt.
  ```
  redirect timing.txt "time_design -post_route"
  ```
- Appends the output to the file test.out.gz in compressed gzip format.
  ```
  redirect test.out.gz "source test.tcl" -append
  ```
- Redirects the output to both screen and a file.
  ```
  redirect timing.txt "time_design -post_route" -append -tee
  ```
- Redirects to the file a.log in compressed gzip format
  ```
  source test.out > a.log.gz
  ```
- Redirects and appends to the file a.log
  ```
  report_area >> a.log
  ```
- Redirects to the Tcl var a
  ```
  report_area > &a
  ```
- Redirects and appends to the Tcl var a
  ```
  report_area >> &a
  ```
- The `get_db` and `get_property` commands return their values as TCL_RESULTS. To redirect the output to any file, you can use any of the following methods:
  a. You can redirect output of `get_db` and `get_property` using the following commands:
  ```
  redirect {puts "[get_db insts .name]"} > inst_name.rpt
  redirect {puts "[get_property [get_lib_cells SDFFHQX1] area]"} > property.rpt
  ```
  b. You can get the value of output stored in a variable using the following commands:
  ```
  redirect a {puts "[get_db insts .name]”} -variable
  redirect b {puts "[get_property [get_lib_cells F1] area]"} -variable
  ```

rename_obj

```
  rename_obj
[-help]
[-escape_ok]
[-flexible]
object
new_name
```

Renames the `base_name` of a single design object, which could only be a `inst`, `hinst`, `design`, `module`, `port`, `hport`, or `hnet`. This command changes only the `base_name` of the specified objects, and cannot move those across the netlist hierarchy. This command also returns the new name of the renamed object.

The names used for SDC timing constraints can only be modified using this command, if they have been loaded into the memory. So, ensure that all the timing views are loaded using the `set_analysis_view` command before running the `rename_obj` command so that the SDC constraints are updated properly. The modified SDC files can be retrieved after renaming the objects.

The `rename_obj` command cannot update the names used inside a CPF `power_intent` file. This is because the CPF usage assumes that the CPF file is golden and should not be modified. To change the names used inside a CPF file, first update the CPF file manually, and then use the updated CPF file to run the `rename_obj` command.

An IEEE1801 `power_intent` file is allowed to be changed during the design flow. So, changing names using the `rename_obj` command will automatically update the IEEE1801 object names. You can use the `rename_obj` command to write out the new IEEE1801 file with the updated names.

Note: To update the inst, net, phys_insts, port, module, and design object names, use the `update_names` command.

Parameters

`-help`	Provides a brief description of the command that includes the type and default information for its parameters. For a detailed description of the command and all its parameters, use the man command: `man rename_obj`

-escape_ok	Allows `new_name` to have either the hierarchy chars '`/`' or the bus-bit chars '`[]`' in it, which will automatically be escaped. If this parameter is not specified, any set of '`/`' or '`[]`' chars in `new_name` will cause an error and the object will not be renamed. *Data_type*: bool, optional
`-flexible`	If `new_name` coincides with an existing name, this parameter appends a suffix (such as `_1` and `_2`) to differentiate it. If this parameter is not specified, a name collision in such a scenario causes an error, and the specified object is not renamed. *Data_type*: bool, optional
object	Specifies the object to be renamed. The object here must be an `inst`, `hinst`, design, `module`, `port`, `hport`, or `hnet`. If *object* is a `port` or `hport`, any `hnet` connected to it with the same `base_name` is also renamed to keep it matched (for example, renaming `hport:top/i1/clkin` also renames `hnet:top/i1/clkin`). If *object* is an `hnet` that is connected to a `port` or `hport` with the same `base_name`, an error occurs. In such a case, you must change the `port` or `hport` name to proceed. If an `hnet` name matches with its associated `net` name (for example, `get_db $my_hnet .net`), then the net name is also renamed on using the `rename_obj` command. For example, renaming `hnet:top/net1` also renames `net:top/net1`. Therefore, to rename a `net`, you need to rename the `hnet` with the same name. This command does not allow renaming a bus-bit object. So any `port`, `hport`, or `hnet` that has a `.base_name` like `busA[0]` (a bus-bit object), cannot be renamed, While a `.base_name` with a scalar name (like `busA\[0\]`) can be changed. *Data_type*: (`inst | hInst | vCell | topCell | term | hTerm | hNet`), required

new_name	Specifies the new `base_name` for the specified object. *Data_type*: string, optional

Examples

- The following command renames an `inst base_name` to `i2_new`:
```
rename_obj inst:top/i1/i2 i2_new
inst:top/i1/i2_new                #returned object with new name
```
- The following command renames an `inst base_name i2\/i3` with escaped hierarchy:
```
rename_obj {inst:top/i1/i2\/i3} i2/i3_new -escape_ok
inst:top/i1/i2\/i3_new            #returned inst object, no hierarchy change
```
- The following command renames a scalar `hport` name to remove escaped bus-bit chars:
```
rename_obj {hport:top/i1/busA\[0\]} busA_0
hport:top/i1/busA_0               #returned hport object
```
Note: `hnet:top/i1/busA\[0\]` is also renamed to `hnet:top/i1/busA_0`.
- The following command renames a `port` name:
```
rename_obj port:top/clkin clkin_new
port:top/i1/clkin_new             #returned port object
```
Note: Both `hnet:top/i1/clkin` and `net:top/i1/clkin` are renamed.

Related Information

`update_names`

report_area

```
  report_area
[-help]
[-detail]
[-hierarchical_instance hinst_name]
[-include_physical]
[-min_area area_per_module]
[-min_count instance_count]
[-out_file filename]
[-show_leaf_cells]
[-show_msv_cells]
[-sort_by {name count area}]
[-table_style {horizontal vertical}]
[-summary | -depth depth_of_hierarchy]
```

Reports the combined standard cell area in each of the hierarchy modules and the top-level design. Also reports the cell area based on different cell types if the `-detail` option is specified. The command provides breakdown based on the cell type (such as .is_* attributes). The report represents the design hierarchy, and therefore, provides better readability.

Note: The units used in the report depend on the units used in the library. The area for each module is calculated from all the standard cells within each module using the LEF information.

Parameters

`-help`	Prints a brief description that includes type and default information for each `report_area` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man report_area`
`-depth depth_of_hierarchy`	Specifies the depth of hierarchy, all levels if not specified; the top of the hierarchy (design) is 1. *Data_type*: int, optional

`-detail`	Report the area in different cell type (buffer, inverter, combinational, flop, latch, clock_gate, macro, physical). *Data_type*: bool, optional
`-hierarchical_instance` *hinst_name*	Reports the area of the leaf instances in the specified hierarchical instance. You can use the -depth option to control the analysis depth of the specified hierarchical instance. *Data_type*: (hinst), optional
`-include_physical`	Specifies to include physical cells in the area calculation and report. By default, it reports only logical cells. *Data_type*: bool, optional
`-min_count` *instance_count*	Specifies the min instance count per module. The generated output will report the area of hinsts with number of instances greater than the specified `min_count value`. *Default*: `0` *Data_type*: int, optional
`-` `min_area` *area_per_module*	Specifies the min area per module. The generated output will report the area of hinsts with area greater than the specified min_area lib_units. *Default*: `0` *Data_type*: float, optional
`-out_file` *filename*	Specifies the name of the output file. *Data_type*: string, optional

`-show_leaf_cells`	Specifies to include the leaf cells at the design level and each hinst level. These are the instances whose parent is the hinst display. *Default*: `0` (Shows the hinst children in the report) *Data_type*: bool, optional
`-show_msv_cells`	Includes multiple supply voltage cells in the report. This option works with "-table_style vertical". *Data_type*: bool, optional
`-sort_by {name count area}`	Specifies the order of hinst names. *Default*: `name` *Data_type*: enum, optional
`-summary`	Prints the area summary for the design. It is equivalent "`-depth 0`". *Data_type*: bool, optional
`-table_style {horizontal vertical}`	Specifies whether the base report should be expanded in the vertical or horizontal direction. *Default*: `horizontal` *Data_type*: enum, optional

Examples

The following command shows the standard cell area in different base classes.

```
report_area -detail
```

Hinst Name	Module Name	Inst Count	Total Area	Combinational	Buffer	Inverter	Flop	Latch	Clock-Gate	Macro
dtmf_recvr_core		4283	37949.153	5223.91	9.526	262.836	2535.221	0.000	0.000	0.000
RESULTS_CONV_INST	results_conv	1331	2201.472	1126.49	0.000	79.38	995.602	0.000	0.000	0.000
TDSP_CORE_INST_MPY_32_INST_csa_tree_mul_61_28	csa_tree	623	1158.419	1115.024	0.000	43.394	0	0.000	0.000	0.000
TDSP_CORE_INST_EXECUTE_INST	execute_i	570	1251.382	504.151	0.000	28.048	719.183	0.000	0.000	0.000
TDSP_CORE_INST_ALU_32_INST	alu_32	466	308.593	742.644	0.000	36.515	0.000	0.000	0.000	0.000
sub_84_22	sub_unsigned_801	93	39.421	170.05	0.000	15.876	0.000	0.000	0.000	0.000
add_81_22	add_unsigned_799	77	28.016	170.755	0.000	0.529	0.000	0.000	0.000	0.000
inc_add_54_27_1	increment_unsigned	31	14.262	107.604	0.000	0	0.000	0.000	0.000	0.000
TDSP_CORE_INST_MPY_32_INST_final_adder_mul_61_28	add_unsigned	74	27.634	165.287	0.000	0	0.000	0.000	0.000	0.000
TDSP_CORE_INST_MPY_32_INST_addinc_add_62_54_8	add_signed_carry	76	28.573	163.523	0.000	0.529	0.000	0.000	0.000	0.000
TDSP_CORE_INST_TDSP_CORE_GLUE_INST_sll_120_49	shift_left_vlog_unsigned_393	92	51.014	128.772	0.000	3.175	0.000	0.000	0.000	0.000
TDSP_CORE_INST_TDSP_CORE_GLUE_INST_sll_121_36	shift_left_vlog_unsigned	56	30.828	94.55	0.000	1.058	0.000	0.000	0.000	0.000

The following command shows the leaf cells at the hinst level.

report_area -show_leaf_cells

```
        Hinst Name                        Module Name              Cell Count  Cell Area  Total Area
-------------------------------------------------------------------------------------------------------
dtmf_recvr_core                                                        4283  37949.153  41732.426
 (Leaf Cells)                                                           995  32001.000  33406.000
 RESULTS_CONV_INST                        results_conv                 1331   2201.472   3311.412
 TDSP_CORE_INST_MPY_32_INST_csa_tree_mul_61_28  csa_tree               623   1158.419   1618.517
 TDSP_CORE_INST_EXECUTE_INST              execute_i                    570   1251.382   1613.524
 TDSP_CORE_INST_ALU_32_INST               alu_32                       466    779.159   1087.752
  sub_84_22                          sub_unsigned_801                93    185.926    225.347
  add_81_22                          add_unsigned_799                77    171.284    200.101
  inc_add_54_27_1                      increment_unsigned            31    107.604    121.866
 TDSP_CORE_INST_MPY_32_INST_final_adder_mul_61_28 add_unsigned        74    165.287    192.921
 TDSP_CORE_INST_MPY_32_INST_addinc_add_62_54_8  add_signed_carry      76    164.052    192.625
 TDSP_CORE_INST_TDSP_CORE_GLUE_INST_sll_120_49  shift_left_vlog_unsigned_393  92  131.947
182.961
```

TDSP_CORE_INST_TDSP_CORE_GLUE_INST_sll_121_36	shift_left_vlog_unsigned	56	95.609	126.436

The following command shows the base report in the vertical direction.

```
report_area -detail -table_style vertical
```

Hinst Name	Module Name	Inst Count	Total Area	
dtmf_recvr_core		4283	37949.15	
(Buffer)		9	9.53	
(Inverter)		491	262.84	
(Combinational)		3279	5223.91	
(Flop)		501	2535.22	
(Latch)		0	0.00	
(Clock-Gate)		0	0.00	
(Macro)		3	29917.66	
RESULTS_CONV_INST	results_conv	1331	2201.47	
(Buffer)		0	0.00	
(Inverter)		150	79.38	
(Combinational)		981	1126.49	
(Flop)		200	995.60	
(Latch)		0	0.00	
(Clock-Gate)		0	0.00	
(Macro)		0	0.00	

report_dangling_ports

```
report_dangling_ports
-help
-partition
-out_file fileName
```

Reports ports that are disconnected from both the inside and outside of a module, and ports that are connected within the module but disconnected outside the module.

Note: If the design has some dangling or unconnected ports, Innovus will add `FE_UNCONNECTED_*` nets for these dangling or unconnected ports in the design during optimization. You can report these dangling ports using the `report_dangling_ports` command.

Parameters

`-help`	Prints a brief description that includes type and default information for each `report_dangling_ports` parameter. For a detailed description of the command and all of its parameters, use the man command: `man report_dangling_ports`
`-out_file`	Specifies the name of the report generated by this command. *Data_type*: string, required
`-partition`	The option a partition only flag of the type boolean. This is an optional. *Data_type*: bool, optional

Examples

The following command writes information on dangling reports to file `myreport`:

```
report_dangling_ports -out_file myreport
```

Related Information

- delete_dangling_ports
- disconnect_dangling_port

report_gate_count

```
report_gate_count
[-help]
[-hinst hinst_name]
[-level level]
[-limit gateCount]
[-std_cell_only]
[-out_file fileName]
```

Reports the size of the imported design, measured in gate counts.

The software calculates the gate count for a module using the following equation:

```
gateCount = moduleArea / gateSize
```

Where:

- `moduleArea` is the area of the module. By default, this is the sum of the areas of all instances inside of the module, including standard cells, blocks, and I/O cells. If you specify the `-std_cell_only` parameter, `moduleArea` is the sum of the areas of the standard cells inside the module only.
- `gateSize` is calculated as follows:

  ```
  gateSize = dbgStdCellHgt x dbgDBUPerIGU x dbgSitesPerGate
  ```

  Where:

  - `dbgStdCellHgt` is the standard cell row height.
  - `dbgDBUPerIGU` is the *M2* layer pitch.
  - `dbgSitesPerGate` is a user-defined global variable that determines the gate size the software assumes when calculating the gate count. For example, the default value of `3` means the assumed the gate size is equal to 1 standard cell row height and 3 *M2* layer pitch widths. You can change this value to adjust the assumed gate size for each specific design.

You can use the `report_gate_count` command after importing the design.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `report_gate_count` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man report_gate_count`.
`-level` *level*	Expands the module to the specified level. For example, if you specify `1`, CTS expands the specified module by one level. The gate counts of all "children" modules of the module that satisfy the `-limit` parameter are included in the gate count report. If you specify `2`, CTS expands the module by two levels. The gate counts of all "children" and "grandchildren" modules of the module are included in the gate count report. *Default*: Module is not expanded, and CTS reports only the gate count of the module itself. (This is equivalent to specifying a level of `0`.) *Data_type*: int, optional
`-limit` *gateCount*	Instructs CTS to ignore modules with gate counts below the specified limit. Such modules are ignored during expansion and reporting. *Default*: CTS ignores modules with 1000 or fewer gates during expansion and reporting. *Data_type*: int, optional
`-hinst` *hinst_name*	Specifies the hierarchical instance for which to generate a gate count report. *Default*: Generates a gate count report for the hierarchical instance corresponding to the top cell of the design. *Data_type*: string, optional
`-out_file` *fileName*	Writes out the gate count report to the specified file name. The default file extension is `.gateCount`. *Default*: Displays the gate count report *only* in the Innovus console. *Data_type*: string, optional

`-std_cell_only`	Calculates gate count based only on standard cells within the specified module. Blocks and I/O cells within the module are ignored. *Data_type*: bool, optional

report_hidden_usage

`report_hidden_usage`
`in_files`

Reports a summary of all the hidden application Tcl globals and mode options in the specified files. Hidden globals and mode options are normally used for temporary fixes or early access, and often disappear or change in a new release. This command will help to find out the hidden settings used in existing scripts that might not be needed in the new release.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `report_hidden_usage` parameter. For a detailed description of the command and all of its parameters, use the man command: `man report_hidden_usage`
`in_files`	Specifies input files to search for hidden usage. *Data_type*: string, required

Examples

- The following command reports the hidden settings in the floorplan.tcl and place.tcl files.

```
>report_hidden_usage {floorplan.tcl cts.tcl} > hidden.rpt

################################################################################
#  Generated by:    Cadence Innovus 18.10-a072_1
#  OS:              Linux x86_64(Host ID sjfib146)
#  Generated on:    Tue Apr 17 22:47:31 2018
#  Design:          top
```

```
#  Command:         report_hidden_usage floorplan.tcl cts.tcl > hidden.rpt
################################################################################


### floorplan.tcl ###
--------------------------------------------------------------------------------
---
Line                       Hidden name                     Command
--------------------------------------------------------------------------------
---
12                        floorplan_xxx                    set_db floorplan_xxx true
32                        init_xxx                         set_db init_xxx 100


### cts.tcl ###
--------------------------------------------------------------------------------
---
Line                       Hidden name                     Command
--------------------------------------------------------------------------------
---
3                          cts_xxx                         set_db cts_xxx true
```

report_messages

```
report_messages
[-help]
[[{-errors | -warnings | -all } [-count ]] | -suppressed ]
[[-prefix prefix [-detail ] [-summary ]] ] [-detail | -count | -suppressed ] [-summary
| -count | -suppressed ]
```

Reports the messages that have been issued since the tool was launched. When you run `report_messages` without any argument, it prints a summary table and returns no Tcl value. However, when you provide an argument, it returns a list a of message IDs and does not print a table.

Parameters

`-help`	Prints a brief description that includes type and default information for each `report_messages` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man report_messages`
`-all`	Returns a list of all issued message IDs as a Tcl result. The result does not include suppressed message IDs. *Data_type*: bool, optional
`-count`	Returns the count of every message when `-errors`, `-warnings`, or `-all` is specified. For example, if an error appeared 20 times, command will return: `{<Error ID> 20}` *Data_type*: bool, optional
`-detail`	Returns a list of detail message of the specified prefix. *Data_type*: bool, optional
`-errors`	Returns a list of issued error message IDs as a Tcl result. *Data_type*: bool, optional
`-prefix` *prefix*	Returns a list of issued error message IDs matched with `-prefix` specified as a Tcl result. *Data_type*: string, optional
`-summary`	Returns the summary of the message that has the specified prefix. *Data_type*: bool, optional
`-suppressed`	Returns a list of suppressed message IDs as a Tcl result. *Data_type*: bool, optional
`-warnings`	Returns a list of issued warning message IDs as a Tcl result. *Data_type*: bool, optional

Examples

- The following command prints a summary table and returns no Tcl value.

```
report_messages
```

Output:

```
*** Summary of all messages that are not suppressed in this session:
Severity           ID               Count              Summary
WARNING          ENCFP-2101         6                Shifter for %s to %s is not
defined in s...
WARNING          ENCFP-3961         1                techSite '%s' has no related
Cells in LE...
WARNING          ENCMSMV-1610       4                View %s domain %s operating
voltage %g d...
WARNING          ENCTS-3            692              Can't create min and max cell
relation f...
WARNING          ENCTS-419          119              Timing arc(s) mismatch found
in cell '%s...
ERROR            ENCTS-423          5                No library found for instance
'%s', cell...
ERROR            ENCTS-424          1                Missing library for some
instance found...
WARNING          ENCEXT-6202        1                In addition to the technology
file, capa...
WARNING          ENCSYT-3046        2                Unknown preference name "%s".
WARNING          ENCSYC-1592        2                No default tech site found for
power dom...
WARNING          ENCPP-4022         2                Option "-%s" is obsolete and
has been re...
WARNING          ENCSC-1001         2                Unable to trace scan chain
"%s". Check t...
WARNING          ENCSC-1020         1                Instance's output pin "%s/%s"
(Cell "%s"...
WARNING          ENCSC-1114         2                The scan chain "%s" is not
traced.
WARNING          ENCSC-1143         1                Unable to apply DEF ordered
sections for...
WARNING          ENCSC-1144         2                Scan chain "%s" was not traced
through. ...
WARNING          ENCOPT-3058        1003             Cell %s/%s has already a
dont_use attrib...
WARNING          ENCOAX-571         1                Property '%s' from OA is a
```

```
hierarchical ...
WARNING          ENCOAX-1257        1              Allowing saveDesign to create
OpenAccess...
WARNING          ENCTCM-77          1              Option "%s" for command %s is
obsolete a...
*** Message Summary: 1843 warning(s), 6 error(s)
```

- The following command reports all issue message IDs.

```
report_messages -all
```

- The following command reports all issue message IDs.

```
report_messages -suppressed
```

- The following command prints the user-defined errors.

```
innovus > report_messages
*** Summary of all messages that are not suppressed in this session:
Severity      ID       Count         Summary
ERROR       no_id       1           A test %s.
ERROR       user        1           A test %s.
WARN        user        1           A test %s.
*** Message Summary: 1 warning(s), 2 error(s)
```

report_netlist_statistics

```
report_netlist_statistics
```

Reports statistical data of the design netlist such as number of cells, number of nets (including tri-nets and degen nets), number of pins, and number of IOs in the design and number nets with different number of terms. The primitives used in the design are also listed.

Example

```
*** Statistics for net list DTMF_CHIP ***
Number of cells      = 4263
Number of nets       = 4551
Number of tri-nets   = 0
Number of degen nets = 21
Number of pins       = 16490
Number of i/os       = 57

Number of nets with    1 terms = 21 (0.5%)
Number of nets with    2 terms = 2510 (55.2%)
Number of nets with    3 terms = 771 (16.9%)
Number of nets with    4 terms = 510 (11.2%)
Number of nets with    5 terms = 223 (4.9%)
Number of nets with    6 terms = 108 (2.4%)
Number of nets with    7 terms = 76 (1.7%)
Number of nets with    8 terms = 46 (1.0%)
Number of nets with    9 terms = 30 (0.7%)
Number of nets with >=10 terms = 256 (5.6%)

*** 72 Primitives used:
Primitive ADDFHX1 (195 insts)
Primitive ADDHX1 (15 insts)
Primitive AND2X1 (279 insts)
Primitive AND3X1 (1 insts)
Primitive AND4X1 (8 insts)
Primitive AOI211X1 (65 insts)
Primitive AOI21X1 (282 insts)
```

report_obj

```
report_obj
[start [pattern] | [-selected ]
[-categories list_of_categories]
[-file fileName]
[-index index_name index_value ...]
[-computed ] [-all ]
[-set_by_user | -read_only]
[-tcl | -verbose ]
```

Creates a report about the attributes for a given pattern/category/object. This command is mainly intended for root attributes but can be used for any object.

It shows the non-default, settable, persistent, not computed, and public attributes of objects.

You can also add more attributes or filter out some attributes using various parameters below.

Parameters

`-help`	Prints a brief description that includes the type and default information for each system root attribute.
`-all`	Reports all non-computed attributes and their current value which could be default or not. *Data_type*: bool, optional
`-categories` *list_of_categories*	Reports only those attributes that are in the given categories. Wildcards * and ? are allowed. Attributes that do not have a category are assigned the reserved category name `NULL`. The report puts all the attributes in a category together. *Data_type*: string, optional
`-computed`	Includes all attributes that are marked as is_computed in the data model. This parameter can be used with `-all`. *Data_type*: bool, optional
`-file` *fileName*	Specifies the output file name. *Data_type*: string, optional

`-index` *`index_name index_value ...`*	Specifies the list of indices in a form of object pointers or pairs of *`obj_type object_name`*, for example. '-index {view view1}' or '-index $obj'. *Data_type*: tcl_obj, optional
`pattern`	Returns all the attribute names that match the specified *`pattern`* using the * and ? wildcards. The *`start`* value is required before the *`pattern`* value. *Default:* * *Data_type*: string, optional
`-read_only`	Reports attributes with `.is_settable = false`. *Data_type*: bool, optional
`start`	Returns attributes for the objects in the *`start`* list. *Default:* `root` *Data_type*: string, optional
`-selected`	Reports attributes for only selected objects. This parameter cannot be used with `-all`. *Data_type*: bool, optional

`-set_by_user`	Reports the root attributes set by the user. In the legacy system, the `setXXXMode` options (e.g. `setPlaceMode`, `setRouteMode`) tracked if the user set them or not independently from the value, while the legacy Tcl globals (e.g. timing_xxx) do not track this. For the root attributes derived from `setXXXMode` options, the `set_by_user` flag can be checked and the root attribute value returned even if the value equals the default value (use `reset_db` to clear the `set_by_user` flag). For root attributes derived from legacy Tcl globals, there is no `set_by_user` flag, so only the root attributes that have a value different from the default value are returned. **Note**: If `-categories` is used along with `-set_by_user`, the attributes in the given categories, as well as all unmapped attributes will be printed. *Data_type*: bool, optional
`-tcl`	Returns the objects and their values in a TCL style list, which is a list of keyword-value pairs. *Data_type*: bool, optional
`-verbose`	Returns the object's data type and default value. This parameter cannot be used with the `-tcl` parameter. *Data_type*: bool, optional

Examples

- The following command displays the root attributes for the `timing` category that are not computed, and do not match the default value with their current values:

```
report_obj -categories timing -all

********************************************************************************** Object:
root:**********************************************************************************

========================================================== Category:
timing==========================================================

Attribute Name                  Current Value
```

---

budget_timing_compare_with_new_opt_data 0budget_timing_push_down_virtual_clock
0budget_timing_write_clock_to_virtual_clock_map 0timing_all_registers_include_icg_cells
truetiming_allow_input_delay_on_clock_source falsetiming_analysis_aocv false

...

- The following command displays all the root attributes that start with `place*`.

```
report_obj / place* -all
********************************************************************************
********
* Object: /
********************************************************************************
********
```

Attribute Name Current Value ----------
-place_blockages {}place_cell_edge_spacing
{}place_design_floorplan_mode falseplace_design_refine_place true...

- The following command displays the attributes that do not match the default value for the net objects in the list:

```
report_obj [get_db nets *clk]
********************************************************************************
********
* Object: net:top/tclk
********************************************************************************
********

Attribute Name Current Value
--------------------------------------------------------------------------------
--------
arcs arc:0x2b1c79348088 arc:0x2b1c793485b0
base_name tclk
bbox {1073741.8235 1073741.8235 -1073741.824 -1073741.824}
...
********************************************************************************
**
* Object: net:top/tclk2
********************************************************************************
**

Attribute Name Current Value
```

```
--------------------------------------------------------------------------------
--
arcs arc:0x2b1c79348088
base_name tclk2
bbox {1073741.8235 1073741.8235 -1073741.824 -1073741.824}
...
```

- The following commands returns the value for the `wire_capacitance*` attribute for the `setup_func_45nm` view.

```
report_obj net:u_core_n_129 wire_capacitance* -index {view setup_func_45nm} -
computed
********************************************************************************
**
* Object: net:bottom_die/u_core_n_129
********************************************************************************
**
Attribute Name                                                          Current
Value
--------------------------------------------------------------------------------
--
wire_capacitance_max                                                  0.0410037
wire_capacitance_max_fall                                             0.0410037
wire_capacitance_max_rise                                             0.0410037
wire_capacitance_min                                                  0.0410037
wire_capacitance_min_fall                                             0.0410037
wire_capacitance_min_rise                                             0.0410037
```

report_obj Category Attributes

The root attributes in the report_obj category are:

- report_obj_display_limit

To set the value of any root attribute, use the `set_db` command:

```
set_db attribute_name value
```

To get the current value of any root attribute, use the `get_db` command:

```
get_db attribute_name
```

To get the current value of all attributes in the `program` category, use the following command:

```
get_db -category program
```

report_obj_display_limit

```
report_obj_display_limit displayLimit
```

Default: `1000`

Data_type: int, read/write

Controls the number of objects that will be displayed when using `report_obj`. If the value is `10`, 'report_obj insts' will only display 10 instance objects with all their non-default attributes.

report_summary

```
report_summary
[-help]
[-out_dir directoryName]
[-no_text | -no_html]
[-out_file fileName | -no_text]
[-no_html | -browser] [-exclude_base_cells_for_density base_cell+ | -
exclude_inst_prefix list_of_prefixes]
```

Reports statistics for the entire design, or a selected object in the design. A report on the entire design includes statistics for the following categories:

- General design information
- General library information

- Netlist information
- Timing information
- Floorplan/Placement information

To display information about a selected object in the design, you must select the object first before using `report_summary`. The information contained in the report varies depending on the object you select. You can select a module, an instance, or a net.

You can use this command at any time after importing a design.

Parameters

`-browser`	Opens a browser and displays the HTML version of the report. *Data_type*: bool, optional
`-exclude_base_cells_for_density` *base_cell+*	Specifies the list of cells to be excluded from the density report. Any instances of these cells are ignored from fixing ((libCell/ptnCell)+. *Data_type*: (base_cell)+, optional
`-exclude_inst_prefix` *list_of_prefixes*	Specifies the list of prefixes to map all instance names to be excluded from the density report. *Data_type*: string, optional
`-no_html`	Generates only a text version of the report. **Note:** If you specify `-no_html`, you cannot specify `-no_text`. *Default:* The report is generated in both HTML and text format. *Data_type*: bool, optional

`-no_text`	Generates only an HTML version of the report. Some statistics in the sections on the main HTML report page include links to more detailed HTML report pages. For example, if you click on the *# Layers* link under the *General Library Information* section of the HTML report, the software displays a detailed HTML report on the layers in the design. **Note:** If you specify `-no_text`, you cannot specify `-no_html`. *Default:* The report is generated in both HTML and text format. *Data_type*: bool, optional
`-out_dir` *directoryName*	Specifies the directory in which the HTML and text reports are saved. If `-no_text` is specified, saves only the HTML version of the report to the specified directory. The `-out_dir` parameter cannot be used with `-no_html`. *Default:* `report_summary` *Data_type*: string, optional
`-out_file` *fileName*	Writes the text version of the report to the specified file. **Note:** You can specify this option only if you specify `-no_html`. *Default:* If you specify `-no_html` and do not specify `-out_file`, the software writes the text version of the report to a file named *design_name*`.main.htm.ascii` in the `report_summary` folder. *Data_type*: string, optional

Examples

Examples

- The following command reports statics for the entire design and generates a text version of the results called `report_summary.rpt`:

  ```
  report_summary -no_html -out_file report_summary.rpt
  ```

  The following is a sample of the resulting text report:

  ```
  ==============================

  General Design Information

  ==============================

  Design Status: Routed

  Design Name: dtmf_chip

  # Instances: 7780

  # Hard Macros:

  ------------------------------

  Macro Cells in Netlist

  ------------------------------

  Macro Name Instance Count Area (um^2) Area Percentage in Core

  ram_256x16A 1 113204.162 26.353%

  rom_512x16A 1 68789.789 16.014%

  pllclk 1 41054.915 9.557%
  ```

```
ram_128x16A 1 100352.611 23.361%

# Std Cells:

------------------------------

Standard Cells in Netlist

------------------------------

Cell Type Instance Count Area (um^2)

XOR2X4 2 113.0976

XOR2X2 7 232.8480

XOR2X1 305 8116.4160

XNOR2X4 8 452.3904

XNOR2X2 8 292.7232

XNOR2X1 146 3885.2352

TLATNX1 1 36.5904

SEDFFX4 8 931.3920

SEDFFX1 124 11136.7872

SDFFSHQXL 1 79.8336

SDFFSHQX4 7 745.1136

.
```

```
.
.

# Pads:

------------------------------

IO Cells in Netlist

------------------------------

I/O Name Instance Count

PCORNERDG 4

PDIDGZ 26

PDO04CDG 27

PVDD1DGZ 5

PVSS1DGZ 5

# Net: 8641

# Special Net: 4

# IO Pins:

------------------------------

Issued IO Information

------------------------------
```

```
# Unplaced IO Pin 0

# Floating IO 0

# IO Connected to Non-IO Inst 0 53

# Pins:

------------------------------

Correctness of Pin Connectivity for All Instances

------------------------------

# Floating Terms 0

# Output Term Marked Tie Hi/Lo 0

# Output Term Shorted to PG Net 0 26897

.
.
.
```

- The following command also reports statics for the entire design, but creates both a text version and an HTML version of the results. Both versions of the report are saved in the `myReport` directory. The text version is named `dtmf_chip.main.htm.ascii` whereas the HTML report is saved as *`design_name`*`.main.htm`.

  ```
  report_summary -out_dir myReport
  ```

Related Information

- For an example of an HTML report, see "HTML Summary Report" in the File Menu chapter of the *Innovus Stylus Common UI Menu Reference*.

resume

```
resume [-help]
```

Resumes a suspended script from the point where it had stopped.

Use the `suspend` and `resume` commands to debug your scripts interactively. The `suspend` command stops a script and restores access to the Innovus prompt. You can then type any command required for debugging. Whenever you want to resume the suspended script, type `resume` at the Innovus prompt.

Parameters

`-help`	Prints the usage of the `resume` command. For a detailed description of the command, use the `man` command: `man resume`.

Related Topics

- `suspend`
- "Interrupting the Software" section of the Getting Started chapter in the *Innovus Stylus Common UI User Guide*.

select_obj

```
select_obj
[-help]
any_object+
```

Selects the object or the list of objects for putting into a set. The object type can be any object type selectable on the GUI such as inst and net.

You can add a new selection to an already selected object using the following command:

```
select_obj [concat [get_db selected.] new_object_dbo_name]
```

Parameters

`-help`	Prints a brief description that includes type and default information for each `select_obj` parameter. For a detailed description of the command and all of its parameters, use the man command: `man select_obj`
`any_object+`	Specifies the list of objects. **Note:** If the list contains the objects that cannot be selected, such as `head`, `layer`, and `ptn`, `select_obj` will not select them. The objects that cannot be selected are `root`, `layer`, `partition`, `bus`, `clock`, `module`, `via_def`, `via_def_rule`, `layer_rule`, `gcell_grid`, `foreign_cell`, `hnet`, `site`, `shape`, `density_shape`, and `track_pattern`. *Data_type*: (any_object)+, required

Example

```
select_obj [get_db insts i1/*]
```

Related Topics

- `deselect_obj`
- Design Implementation Flow chapter in the *Innovus Stylus Common UI User Guide*

set_compress_level

`set_compress_level` *`compression_level`*

Specifies the compression level to use when compressing large data files. A higher compression level creates a more compressed file, but at the cost of speed. A lower compression level creates a less compressed file, but speeds up the process. By default, the software uses the `gzip` default value, that is 6, which results in high compression at the cost of speed.

Parameters

`-help`	Prints a brief description that includes type and default information for each `set_compress_level` parameter. For a detailed description of the command and all of its parameters, use the man command: man `set_compress_level`
`compression_level`	Specifies the compression level to use when compressing files. You can specify -1 or values between 1 and 9, where: `1` Creates the least compressed files at the highest speed. `9` Creates the most compressed files at the lowest speed. -1 Returns to the `gzip` default value, that is 6 (`Z_DEFAULT_COMPRESSION`). *Data_type*: int, required

Examples

- The following command sets the highest compression level and results in the most compressed data:
  ```
  set_compress_level 9
  ```
- The following command resets the compression level to the gzip default, that is 6.
  ```
  set_compress_level -1
  ```

Related Commands

- get_compress_level

set_current_design

`set_current_design` *`cell_name`*

Switches the design to the partition specified by the `top` *`cell_name`*.

Note: Cadence recommends that you use the same name for the top cell and the partition.

You can use this command only after you commit the partition.

Parameters

`cell_name`	The name of the top cell that indicates a different partition. *Data_type*: string, required

Related Topics

- Hierarchical Flow Capabilities chapter in the Innovus User Guide
  - Partitioning the Design - Restoring the Top-Level Floorplan with Partition Data

set_layer_preference

```
set_layer_preference
[-help]
layer_name
[-is_visible {1 | 0}]
[-is_selectable {1 | 0}]
[-color {color_name | color_value}]
[-line_width {lw}]
[-stipple stipple_name | -stipple_data width height "stipple_data"]
```

Sets a new preference for the specified object or layer.

See `get_layer_preference` for descriptions of how to display layer preferences.

Parameters

`-help`	Prints a brief description that includes type and default information for each `set_layer_preference` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man set_layer_preference`
`-color {color_name | color_value}`	Specifies a color for the object. Specify either a color name or a numeric RGB value. *Data_type*: string, optional
`-is_selectable {1 | 0}`	Specifies whether the object is selectable. Specify `1` to make the object selectable. Specify `0` to make the object non-selectable. *Data_type*: enum, optional

`-is_visible {1 | 0}`	Specifies whether the object is visible. Specify `1` to make the object visible. Specify `0` to make the object invisible.
	Examples
	• You can turn off the visibility of all custom layers by using the following command:
	`set_layer_preference customLayers -is_visible 0`
	• You can turn off display of flightlines by using the following command:
	`set_layer_preference flightLine -is_visible 0`
	Data_type: enum, optional

layer_name	Specifies the name of the layer or the object type. The valid layer names are `M#`, where `#` is `0` through `1`, and `M#AVio`, `M#Cont`, `M#ContPin`, `M#ContRB`, `M#Pin`, `M#RB`, `M#RTrk`, and `M#Vio`. The valid object types are `blackBox`, `block`, `blockHalo`, `bump`, `bump1`, `bump2`, `bump3`, `cellBlkgObj`, `channel`, `clkTree`, `congest`, `congestH`, `congestObj`, `congestV`, `datapath`, `fence`, `gcellOvflow`, `guide`, `hinst`, `inst`, `ioSlot`, `ioRow`, `macroSitePattern`, `metalFill`, `mGrid`, `net`, `netRect`, `nonPrefTrackObj`, `obstruct`, `pinObj`, `pinText`, `power`, `powerNet`, `ptnFeed`, `ptnPinBlk`, `pwrdm`, `pwrdm1`, `pwrdm2`, `pwrdm3`, `layerBlk`, `region`, `routeGuide`, `ruler`, `screen`, `select`, `sitePattern`, `stdRow`, `term`, `text`, `trackObj`, `userGrid`, and `violation`. *Data_type*: string, required
`-line_width` *lw*	Specifies the number of pixels in the border of the specified object. Specify this value as integer. *Data_type*: int, optional

`-stipple` *stipple_name*	Specifies information about the stipple pattern of the object. The valid stipple names are `None`, `Horizontal`, `Vertical`, `Grid`, `Slash`, `Backslash`, `Cross`, and `Brick`. *Data_type*: string, optional
`-stipple_data` *width height "stipple_data"*	Specifies the stipple information for the object. Specify this information in x-window bitmap format. *Data_type*: string, optional

Example

The following command sets a layer preference for the module object in green with dotted stipple pattern (8 by 4 bitmap with dot at every 4 pixels):

```
set_layer_preference hinst -color green -stipple_data 8 4 "0x00 0x11 0x00 0x00"
```

Related Information

- `get_layer_preference`

set_license_check

```
set_license_check
[-help]
[-option_list {license1 license2 …}]
[-wait time_in_minutes]
[-default]
[-check_out license]
[-status]
[-exists_on_server]
```

Manages licenses for product options. Specifies licenses to check out, license wait time, and the status of all product option licenses.

Note: This command does not manage multi-CPU licenses. For information on the commands that manage multi-CPU licenses, see the "Multiple-CPU Processing Commands" chapter.

Parameters

`-check_out` *license*	Checks out the licenses for the specified product options when this command runs. *Data_type*: string, optional
`-default`	Restores the default. values for this command. *Data_type*: bool, optional
`-exists_on_server`	Outputs a table that provides the following information for each of the licenses that exist on the license server. • License name • Product number • Product name • License string • Version *Data_type*: bool, optional

`-help`	Outputs a brief description that includes type and default information for each `set_license_check` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man set_license_check`.
`-option_list` *license1 license2 …*	Specifies licenses to check out dynamically. Dynamic licenses are not checked out until they are needed. If you specify a license that is not allowed with your base license, or a license that is not available, the software does not check out that license and instead issues a warning message. If you specify more than one license, begin and end the option list with double quotation marks or braces. *Data_type*: string, optional
`-status`	Outputs the following information: • The name of the base license that is checked out • The current wait time • A list of the allowed license options • A list of the available license options • A list of the currently checked-out license options. *Data_type*: bool, optional
`-wait` *time_in_minutes*	Specifies the amount of time the system waits for a license to become available. If the license is available in less than the specified wait time, the system checks out the next needed license without waiting. *Default:* `0` (no wait time) *Value range:* `0` to `10,000` *Data_type*: int, optional

Examples

- The following command outputs a table that provides information about the licenses that exist on the license server:

  ```
  set_license_check -exists_on_server
  ```

  The system outputs the following information:

  ```
  Name         Prod #    Product Name  License string                          Version
  invs_hier    INVS40    Innovus Hierarchical Design Option Innovus_Hier_Opt  15.1
  invs_ms      INVS30    Innovus Mixed Signal Option Innovus_MS_Opt           15.1
  invs_3d_ic   INVS60    Innovus 3D-IC Option Innovus_3D_IC_Opt               15.1
  invs_20nm    INVS20    Innovus 20nm Option Innovus_20nm_Opt                 15.1
  invs_10nm    INVS10    Innovus 10nm Option Innovus_10nm_Opt                 15.1
  invs_i20nm   INVSI20   Innovus I20nm Option Innovus_I20nm_Opt               15.1
  invs_i10nm   INVSI10   Innovus I10nm Option Innovus_I10nm_Opt               15.1
  ................................................
  ................................................
  ................................................
  ```

- The following command specifies that licenses for Tempus-L and Tempus-XL may be checked out. Tempus-L licenses are checked out before Tempus-XL licenses.

  ```
  set_license_check -option_list {tpsl tpsxl}
  ```

- The following command sets the wait time to 5 minutes:

  ```
  set_license_check -wait 5
  ```

Related Topics

- Multiple-CPU Processing Commands
- Product and Licensing Information chapter in the *Innovus Stylus Common UI User Guide*
- Getting Started chapter in the *Innovus Stylus Common UI User Guide*

set_limited_access_feature

```
set_limited_access_feature
[-help]
featureName 1 | 0
```

Enables or disables a limited access feature.

Innovus includes certain limited access features that showcase advanced technology integrated in the software. These features have been internally qualified at Cadence but have had only limited customer testing.

> Limited access features are enabled by a variable specified through the `set_limited_access_feature` command. To use these limited access features, contact your Cadence representative to qualify your usage and to make sure it meets your needs before deploying it widely.

Parameters

`-help`	Prints a brief description that includes type and default information for each `set_limited_access_feature` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man set_limited_access_feature`
`featureName`	Specifies the name of the limited access feature. *Data_type*: string, required
*`1	0`*	Specifies the name of the variable that enables the feature. Specifying a value of `1` enables the feature. Specifying a value of `0` disables the feature. *Data_type*: bool, required

set_log_post_cmd

```
set_log_post_cmd
[-help]
[-disable command_list]
[-enable command_list]
[-report]
```

Turns on/off logging specific commands inside a source file that would be logged in the post-expansion mode (suc as source_verbose = true and log_verbose_type = post). It can be used to suppress commands, such as ‘set’ and ‘get_db’ to reduce clutter in a large post-expansion log file.

Note: This command does not affect pre-expansion logging.

Parameters

`-help`	Prints a brief description that includes type and default information for each `set_log_post_cmd` parameter. For a detailed description of the command and all of its parameters, use the man command: `man set_log_post_cmd`
`-disable` *command_list*	Specifies a a list of command names to not log. *Data_type*: string, optional
`-enable` *command_list*	Specifies a a list of command names to be logged. *Data_type*: string, optional
`-report`	Writes out each disabled commands to stdout. One command is written per line. *Data_type*: bool, optional

Related Information

- `get_db`

set_message

```
  set_message
-help
-id list_of_msgIDs
[-severity {warn error info} | -limit <number> | -no_limit | -suppress | -unsuppress |
-on_error {msg_only exit stop_script} | -stop_script_limit limitNumber]
```

Specifies to change the message severity levels, `INFO, WARNING` or `ERROR`, at the beginning of the message. This command allows you to control the messages that are displayed for a particular design. For example, if there are some warnings that are critical for the design, you can change the message severity to `ERROR`. Similarly, if there are messages that you want to ignore, you can change the message severity to `INFO`.

An example of an ERROR message is:

**ERROR: (GLOBAL-105): Unknown global 'timing_enable_socv_analysis' specified.

All ERROR and WARN message begin with a header like:

**ERROR: (<msg_id>):

**WARN: (<msg_id>):

The <msg_id> format is: <prefix>-<num> ("GLOBAL" is the prefix, and "105" is the num in the example above).

You can use the msg_id to change how that specific message is handled using the options below.

Parameters

-help	Outputs a brief description that includes type and default information for each `set_message` parameter. For a detailed description of the command and all of its parameters, use the man command: man `set_message`.
`-id` *list_of_msgIDs*	Specifies the list of message IDs that require a change in the message severity level. If you do not specify a message ID, the change in severity level is applied to all message IDs. *Data_type*: string, optional

`-limit` *`number`*	Sets the message limit to *`number`* per every public command. The default limit is 20, so any specific message will be written out 20 times for one command and then further output of that message will be disabled (along with an additional message saying the message limit has been exceeded). If the number is set to 0, the message will be completely suppressed for the next public command that triggers the message. You can unsuppress the message by setting the number greater than 0. *Data_type*: int, optional
`-no_limit`	Removes any existing limit on the selected messages. **Note:** Suppressed messages remain suppressed until the `-unsuppress` option is applied. *Data_type*: bool, optional
`-on_error {msg_only exit stop_script}`	Specifies an extra action for error messages (not for warn or info messages). If no message is specified, the action applies to all error messages. • `msg_only`: Prints error message only, and the command proceeds as it normally would. This is the default. • `exit`: Exits immediately and returns to Linux. • `stop_script`: Waits for the current Tcl command to finish, then stops executing any Tcl script, and returns to the interactive Tcl prompt. Note, if a message severity is changed from error to either warn or info, it will no longer trigger this extra action. *Data_type*: enum, optional
`-severity {warn error info reset}`	Specifies the new severity level for the message ID being modified. The `reset` argument sets the message severity back to its original state. *Data_type*: enum, optional

`-stop_script_limit` *limitNumber*	Sets the message limit for a public command. If you set `-stop_script_limit` for a command as `10` and message for the command is triggered for the 10th time, the tool will wait for the current Tcl command to finish, then stop executing any Tcl script, and return to the interactive Tcl prompt. The default value of `-stop_script_limit` is `-1`, which is unlimited.
`-suppress`	Suppresses the message. *Data_type*: bool, optional
`-unsuppress`	Unsuppresses the message. *Data_type*: bool, optional

Examples

- The following command changes the message IMPLF-81 from `ERROR` to `WARN`:
  ```
  set_message -id IMPLF-81 -severity warn
  ```
- The following command returns all messages to their original severity:
  ```
  set_message -severity reset
  ```
- The following command resets the specified message to its original severity:
  ```
  set_message -id ENCLF-18  -severity reset
  ```

Related Information

- `get_message`

set_preference

```
set_preference
[-help]
preference_name
value
```

Defines preferences, such as, commands issued during a session can be logged to a log or the screen.

When writing to a log and screen, the commands are preceded by <CMD>.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `set_preference` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man set_preference`.
`preference_name`	The name of the preference variable you want to set. To see a list of preferences, select ***View - Set Preference*** from the menu, then click *Save*. In the Save Preferences form, select the required save option and click *OK*. If you select the default *Save to Home Directory* option, the list of preferences and their values are saved in `.enc` file in your home directory. *Data_type*: string, required
`value`	The value for the preference. *Data_type*: string, required

Preference Variables

The following table lists the preference variables you can set using the `set_preference` command and their GUI equivalent in the Preferences form:

Note: The following table might not be readable when viewed with the Unix man command. Check the `set_preference` command description in the *Innovus Text Command Reference* to view the table.

Variable Name	GUI Equivalent in Preferences Form	
	Tab Name	Option Name
DesignName	***Design***	*Design Name*
DesignHierChar		*Design Hierarchical Character*
DEFHierChar		*DEF Hierarchical Character*
PDEFBusDelim		*PDEF Bus Delimiter*
LogTypeInCmd		*Log Type-In Command*
logviewer		*Invoke viewLog at Start Up*
MouseGestures		*Enable Mouse Gestures*
ShowFPObjInPlace	***Display***	*Show Floorplan Objects in Physical View*
ShowAllFence		*Show All Constraint*
DisplayPtnPin		*Display Partition Pin*
ShowAttrPopup		*Show Attributes Popup*
ShowUnplacInnovusnst		*Display Unplaced Instances*
ShowRectilinearPad		*Show Rectilinear Pad Cell*
DisplayCellPin		*Display Instance Pin*
ShowLefLayerName		*Show Lef Layer Name*
AutoRedraw		*Auto-redraw after Layer Change*
EnlargeLogicalPin		*Enlarge Logical Pin In Fit View*
InstDisplayThreshold		*Instance Display Threshold*
WirInnovussplayThreshold		*Wire Display Threshold*
WirInnovussplayCheck2D		*Apply Threshold in 2D*
percentageOfPan		*Percentage of Window to Pan*

`scaleOfZoom`		*Scale of Zoom In/Out*
`zoomPrevCount`		*Number of Previous Zoom Saved*
`MinFPModuleSize`		*Min. Floorplan Module Size*
`ShowParentModule`		*Show Parent Module With Level*
`ShowChildPartition`		*Show Child Partition With Level*
`ShowParentPartition`		*Show Parent Partition With Level*
`AutoDetailDisplay`		*Automatic Detail Display for Large Design* ◦ Special Wire ◦ Regular Wire
`DetailDisplayFactor`		*Factor*
`DisplayRelFPlan`	***Innovus***	*Honor Relative Floorplan Constraints*
`SingleLayerMode`		*Single Layer Mode For Routing Blockage, Ptn Pin Blockage and Ptn Feed*
`ShowDelBox`		*Show Delete Confirmation Box*
`StrecthRestriction`		*Box Stretch Restrictions* ◦ *No Restriction* ◦ *Maintain Area* ◦ *Maintain Aspect Ratio*
`StretchRectilinear`		*Rectilinear Stretch Restrictions* ◦ *No Restriction* ◦ *Maintain Area*
`MoveRestriction`		*Move Restrictions* ◦ *No Restrictions* ◦ *Orthogonal*

clone_row_orientation_snapping		*Clones snapping to same master row orientation*
SelectByArea	***Selection***	*Select by Area* ◦ *Enclosed by Box* ◦ *Intersecting Box*
SelectByLine		*Select by Line*
SelectStickyMode		*Sticky Mode*
SelectNetWhenSelectPin		*Hilite Net When Selecting a Pin*
HiliteNetWhenQueryObj		*Hilite Nets When Querying an Instance*
HiliteShapeWhenSelectNet		*Hilite Pin Shapes When Selecting a Net*
QueryWireNet		*Query Nets When Wires are Selected*
QuerySkipInst		*Skip Objects by Auto Query* ◦ *Cell*
QuerySkipInstObs		◦ *Obs in Cell*
QuerySkipInstPin		◦ *Pin in Cell*
QuerySkipRegular		◦ *Regular Wire*
QuerySkipSpecial		◦ *Special Wire*
WinSelectMargin		*Window Select Margin*
HighlightColorNumber		*Number of Highlight Color*
MinFlightLineWidth	***Flightline***	*Minimum Flightline Width*
MaxFlightLineNetTerms		*Maximum Flightline Net Terms*
DisplayPlaceFlightLine		*Show Flightline In Physical View*
ShowNumberBlockConnection		*Show Number of Connections For Block*

FlightLineInMove	*Draw Flightline When Moving Instance(s)*
ShowConnectionInOutNumber	*Show Input/Output Connection Number*
ShowFlightLineTermMark	*Show Pin's Input/Output Mark*
ShowBothInputConnection	*Show Both Input Pins Connection*
ShowNetWeightConnection	*Show Connection With Netweight 0*
ShowFlightLineThruPtnPin	*Show Flightlines Through Partition Pins*
noInsideMacro	*Don't Show Flightline to Inside Macro*
SkipBufferFlightline	*Skip Buffer/Invert Cell*
NoClockFlightline	*Don't Show Clock Net*
OnlyClockFlightline	*Show Clock Net Only*
DrawFlightlineLast	*Draw Flightline on Top of Selection*
OnlyBundleNetFlight	*Only Show Net Bundle Flight Line*
ExclusiveFlight	*Show Connection Only Between Selected Objects*
ConnectionColorType	*Single Color* *Different Color by Type of Connections* *Different Color by Number of Connections*
InputConnectionColor	• *Different Color by Type of Connections: Input* • *Different Color by Number of Connections: Small*
OutputConnectionColor	• *Different Color by Type of Connections: Output* • *Different Color by Number of Connections: Medium*

`InoutConnectionColor`	• *Different Color by Type of Connections: Inout* • *Different Color by Number of Connections: Large*
`MixtureConnectionColor`	• *Single Color: <color>* • *Different Color by Type of Connections: Inout*
`ShowConnectionWithWidth`	*Different Width by Number of Connections*
`FLWidthThresholdLow`	*Number of Connections* ◦ *Small* - Fewer than specified low threshold
`FLWidthThresholdHigh`	◦ *MInnovusum* - Equal to or between than specified low and high thresholds ◦ *Large* - Greater than specified high threshold
`NoInstFlightLine`	*Don't Show Connection to Unselected Object* ◦ *Instance*
`NoBlockFlightLine`	◦ *Block*
`NoModuleFlightLine`	◦ *Module*
`NoBlackBlobFlightLine`	◦ *Black Blob*
`NoBlackBoxFlightLine`	◦ *Black Box*

InstanceText	***Text***	*Instance Text* ◦ *Instance Name* ◦ *Master Name*
ShowNetFullName		*Show Full Net Name*
ShowNetNameWithLayerColor		*Show Net Name With Layer Color*
ShowModuleText		*Object Text Display* ◦ *Module*
ShowAmoebaModuleText		◦ *Module in Amoeba*
ShowRowSiteText		◦ *Row Site*
ShowIoPadText		◦ *IO Pad*
ShowInstanceText		◦ *Instance*
ShowInstancePinText		◦ *Instance Pin*
ShowBlackBlobText		◦ *Black Blob*
ShowIoPinText		◦ *IO Pin*
ShowGroupText		◦ *Group*
ShowBumpText		◦ *Bump*
ShowClockTreeText		◦ *Clock Tree*
ShowChannelText		◦ *Channel*
ShowLefPortNumText		◦ *Lef Port Num*

ShowMacroSitePtnText		◦ *Macro Site Ptn*
ShowSIPFingerText		◦ *SIP Finger*
ShowNetText		◦ *Net*
TextDisplaySize		*Text Display Size* ◦ Small ◦ MInnovusum ◦ Large ◦ Auto
TermCrossPix		*Term Cross Symbol Size*

Note: Options on the ***Windows*** tab of the Preferences form map to the `set_window_preference` command instead of the `set_preference` command.

The following table lists the preference variables you can set using the `set_preference` command and their type, range/default/sample value.

Note: The following table might not be readable when viewed with the Unix man command. Check the `set_preference` command description in the *Innovus Text Command Reference* to view the table.

Variable Name	Type	Range/Default/Sample Value (Default values in bold)
DesignName	String	/spc/spc4/FE_tests_7.1/designs../dtmf_pso/dtmf_fpd.enc.dat
DesignHierChar	String	**/**
DEFHierChar	String	**/**
PDEFBusDelim		**[]**
LogTypeInCmd	Boolean	**0 – False** 1 – True

logviewer	Boolean	**0 – False** 1 – True
MouseGestures	Boolean	0 – False **1 – True**
ShowFPObjInPlace	Boolean	**0 – False** 1 – True
ShowAllFence	Boolean	**0 – False** 1 – True
DisplayPtnPin	Boolean	0 – False **1 – True**
ShowAttrPopup	Boolean	0 – False **1 – True**
ShowUnplacInnovusnst	Boolean	**0 – False** 1 – True
ShowRectilinearPad	Boolean	**0 – False** 1 – True
DisplayCellPin	Boolean	**0 – False** 1 – True
ShowLefLayerName	Boolean	0 – False **1 – True**
AutoRedraw	Boolean	0 – False **1 – True**
EnlargeLogicalPin	Boolean	**0 – False** 1 – True
InstDisplayThreshold	Real	**0.0**

WirInnovussplayThreshold	Integer	**0**
WirInnovussplayCheck2D	Boolean	0 – False **1 – True**
percentageOfPan	Enum	10, 20, 30, 40, 50, 60, 70, 80, 90, **95**, 100
scaleOfZoom	Real	**2.0**
zoomPrevCount		**1** to 6
MinFPModuleSize	Integer	**100**
ShowParentModule	Integer	**0**
ShowChildPartition	Integer	**0**
ShowParentPartition	Integer	**0**
AutoDetailDisplay	Boolean	**0 – False** 1 – True
DetailDisplayFactor	Integer	**16**
DisplayRelFPlan	Boolean	0 – False **1 – True**
SingleLayerMode	Boolean	**0 – False** 1 – True
ShowDelBox	Boolean	**0 – False** 1 – True
StrecthRestriction	Enum	**0 – No Restriction** 1 – Maintain Area 2 – Maintain Aspect Ratio
StretchRectilinear		**0 – No Restriction** 1 – Maintain Area

MoveRestriction		**0 – No Restrictions** 1 – Orthogonal
clone_row_orientation_snapping	Boolean	**0 – False** 1 – True
SelectByArea	Boolean	**0 – Enclosed by Box** 1 – Intersecting Box
SelectByLine	Boolean	**0 – False** 1 – True
SelectStickyMode	Boolean	**0 – False** 1 – True
SelectNetWhenSelectPin	Boolean	**0 – False** 1 – True
HiliteNetWhenQueryObj	Boolean	0 – False **1 – True**
HiliteShapeWhenSelectNet	Boolean	**0 – False** 1 – True
QueryWireNet	Boolean	**0 – False** 1 – True
QuerySkipInst	Boolean	**0 – False** 1 – True
QuerySkipInstObs	Boolean	**0 – False** 1 – True
QuerySkipInstPin	Boolean	**0 – False** 1 – True
QuerySkipRegular	Boolean	**0 – False** 1 – True

QuerySkipSpecial	Boolean	**0 – False** 1 – True
WinSelectMargin	Integer	**8**
HighlightColorNumber	Enum	**64**, 128, 256
MinFlightLineWidth	Integer	**1**
MaxFlightLineNetTerms	Integer	**500**
DisplayPlaceFlightLine	Boolean	0 – False **1 – True**
ShowNumberBlockConnection	Boolean	**0 – False** 1 – True
FlightLineInMove	Boolean	0 – False **1 – True**
ShowConnectionInOutNumber	Boolean	**0 – False** 1 – True
ShowFlightLineTermMark	Boolean	**0 – False** 1 – True
ShowBothInputConnection	Boolean	**0 – False** 1 – True
ShowNetWeightConnection	Boolean	0 – False **1 – True**
ShowFlightLineThruPtnPin	Boolean	**0 – False** 1 – True

noInsideMacro	Boolean	**0 – False** 1 – True
SkipBufferFlightline	Boolean	**0 – False** 1 – True
NoClockFlightline	Boolean	**0 – False** 1 – True
OnlyClockFlightline	Boolean	**0 – False** 1 – True
DrawFlightlineLast	Boolean	**0 – False** 1 – True
OnlyBundleNetFlight	Boolean	**0 – False** 1 – True
ExclusiveFlight	Boolean	0 – False **1 – True**
ConnectionColorType	Enum	**0 – Single Color** 1 – Different Color by Type of Connections 2 – Different Color by Number of Connections
InputConnectionColor		**green**
OutputConnectionColor		**yellow**
InoutConnectionColor		**pink**
MixtureConnectionColor		**blue**
ShowConnectionWithWidth	Boolean	**0 – False** 1 – True

FLWidthThresholdLow	Integer	**20**
FLWidthThresholdHigh	Integer	**80**
NoInstFlightLine	Boolean	**0 – False** 1 – True
NoBlockFlightLine	Boolean	**0 – False** 1 – True
NoModuleFlightLine	Boolean	**0 – False** 1 – True
NoBlackBlobFlightLine	Boolean	**0 – False** 1 – True
NoBlackBoxFlightLine	Boolean	**0 – False** 1 – True
InstanceText		**Instance** Master
ShowNetFullName	Boolean	**0 – False** 1 – True
ShowNetNameWithLayerColor	Boolean	**0 – False** 1 – True
ShowModuleText	Boolean	0 – False **1 – True**
ShowAmoebaModuleText	Boolean	0 – False **1 – True**
ShowRowSiteText	Boolean	0 – False **1 – True**

ShowIoPadText	Boolean	0 – False **1 – True**
ShowInstanceText	Boolean	0 – False **1 – True**
ShowInstancePinText	Boolean	0 – False **1 – True**
ShowBlackBlobText	Boolean	0 – False **1 – True**
ShowIoPinText	Boolean	0 – False **1 – True**
ShowGroupText	Boolean	0 – False **1 – True**
ShowBumpText	Boolean	0 – False **1 – True**
ShowClockTreeText	Boolean	0 – False **1 – True**
ShowChannelText	Boolean	0 – False **1 – True**
ShowLefPortNumText	Boolean	0 – False **1 – True**
ShowMacroSitePtnText	Boolean	0 – False **1 – True**
ShowSIPFingerText	Boolean	0 – False **1 – True**

ShowNetText	Boolean	0 – False **1 – True**
TextDisplaySize	Enum	**s**, m, l, a
TermCrossPix	Enum	**4**, 6, 8

Related Topics

For more information on setting preferences from the GUI, see:

- get_preference
- Set Preference form in the View Menu chapter in the *Innovus Stylus Common UI Menu Reference*.

set_proc_verbose

```
set_proc_verbose
[-help]
{[procedure_name [-quiet ]] | [-report ]}
```

Makes a procedure verbose, when the procedure is called for execution.

Enables verbose logging of the commands inside a specific Tcl procedure. For example, if you make ‘proc test’ verbose, and type ‘test’ at the command line, you would see something like:

```
@innovus 1> test

#@ Begin verbose proc (pre): test

@proc 1: set a 1

@proc 2: puts "inside test with $a"

inside test with 1

#@ End verbose proc: test
```

Parameters

`-help`	Outputs a brief description that includes type and default information for each `set_proc_verbose` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man set_proc_verbose`.
`procedure_name`	Specifies the name of the procedure to be made verbose. *Data_type*: string, required
`-quiet`	Changes the procedure back to not being verbose. *Data_type*: bool, optional
`-report`	Reports procedures that are verbose. *Data_type*: bool, optional

Examples

- The following commands show the usage of the `set_proc_verbose` command and its options for the `fplan_boxes` procedure.

  The code within the `fplan_boxes` procedure is given below:

  ```
  @innovus 1> proc fplan_boxes {} {

  + puts "Floor Plan Boxes:"

  + puts [get_db current_design .boundary]

  + }
  ```

  The following command executes the `fplan_boxes` procedure. Please note that the execution result simply shows the coordinates of the floor plan boxes.

  ```
  @innovus 2> fplan_boxes

  Floor Plan Boxes:

  {{93.818 -645.458} {93.818 -639.731} {102.843 -639.731} {102.843 -645.458}}
  ```

  The following command makes the `fplan_boxes` procedure verbose:

  ```
  @innovus 3> set_proc_verbose fplan_boxes

  fplan_boxes
  ```

  After the `fplan_boxes` procedure is made verbose and is executed, the code of the `fplan_boxes` procedure is displayed:

  ```
  @innovus 4> fplan_boxes

  #@ Begin verbose proc fplan_boxes
  @proc(fplan_boxes) 1: puts "Floor Plan Boxes:"
  Floor Plan Boxes:
  @proc(fplan_boxes) 2: get_db current_design .boundary
  @proc(fplan_boxes) 3: puts [get_db current_design .boundary]
  {{93.818 -645.458} {93.818 -639.731} {102.843 -639.731} {102.843 -645.458}}
  #@ End verbose proc fplan_boxes
  ```

  You can use the `-report` option to see the verbose procedures:

  ```
  @innovus 6> set_proc_verbose -report
  fplan_boxes
  ```

  If you have made your procedure verbose and now you want to make it non-verbose, you can use the `-quite` option. The command below makes the `fplan_boxes` procedure non-verbose:

  ```
  @innovus 6> set_proc_verbose fplan_boxes -quiet
  ```

```
fplan_boxes
```

set_window_preference

```
set_window_preference
[-help]
-window_name
{on | off}
```

Enables or disables windows (panels) from the Innovus main display. You can save these window settings in the .enc file in your home directory. The settings are automatically loaded when invoking the Innovus main window.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `set_window_preference` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man set_window_preference`.
`-window_name`	Specifies the name of the window you want to turn on or off in the Innovus main display. You can select from the following: • `properties` -- Controls the Attribute Viewer panel. • `status` -- Controls the message bar display, which shows object identifier or coordinate information. • `toolbox` -- Controls the tool widget display. • `viewColor` -- Controls the visibility, selectability, and color display of various attributes. • `world` -- Controls the satellite window display. *Data_type*: string, required
`{on | off}`	Specifies whether window should be turned on or off. *Data_type*: string, required

Example

The following command turns on the tool widget display:

```
set_window_preference toolbox on
```

source

```
source
[-help]
filename
[-quiet | verbose]
[-quiet  | -echo ]
```

Reads a file and executes the commands in the file (scripts).

Parameters

`-help`	Describes the command usage.
`-echo`	Enables verbose source. *Data_type*: bool, optional
`filename`	Specifies the name of the file to source. *Data_type*: string, required
`-quiet`	Forces source to be normal Tcl source command. *Data_type*: bool, optional
`-verbose`	Enable verbose source. *Data_type*: bool, optional

With no `-quiet, -verbose, -echo` argument, file is sourced according to the `source_verbose` attribute.

Note that the setting of `-quiet, -verbose, -echo` argument only affects the verbosity of sourcing the outermost file. If that file contains nested `source` commands, the verbosity of those nested commands is controlled by their associated `[-quiet | -verbose | -echo]` arguments or the `source_verbose` attribute if no arguments.

suspend

```
suspend [-help]
```

Suspends your script and returns to the Innovus prompt.

The `suspend` command gives you more flexibility in managing and debugging your scripts. You can insert multiple `suspend` commands in a script. The `suspend` command can also be used in nested scripts. When the script reaches a point where you have inserted the `suspend` command, it stops and restores access to the Innovus prompt. You can then type any command required for debugging.

Whenever you want to resume your script, just type `resume` at the Innovus prompt.

Parameters

`-help`	Prints the usage of the `suspend` command. For a detailed description of the command, use the `man` command: `man suspend`.

Related Information

- `resume`
- "Interrupting the Software" section of the Getting Started chapter in the *Innovus Stylus Common UI User Guide*

system Category Attributes

The root attributes in the system category are:

- auto_file_dir
- auto_file_prefix
- distributed_mmmc_disable_reports_auto_redirection
- gui_verbose
- layer_name_no_abbreviation
- log_verbose_type
- print_full_message_summary
- read_db_directory
- read_db_file_check
- read_db_stop_at_design_in_memory
- read_db_tool_name
- read_db_version
- script_search_path
- source_echo_filename
- set_db_verbose
- soft_stack_size_limit
- source_verbose
- source_continue_on_error
- log_verbose_style
- source_verbose_line_length_limit
- tcl_error_info
- tcl_partial_cmd_argument_matching
- tcl_return_display_length_limit
- ui_precision

- ui_precision_capacitance
- ui_precision_derating
- ui_precision_power
- ui_precision_sensitivities
- ui_precision_timing
- write_db_auto_save_user_globals
- write_db_cmd_file_limit

To set the value of any root attribute, use the `set_db` command:

```
set_db attribute_name value
```

To get the current value of any root attribute, use the `get_db` command:

```
get_db attribute_name
```

To get the current value of all attributes in the `system` category, use the following command:

```
get_db -category system
```

auto_file_dir

`auto_file_dir` *directoryName*

Default: ./

Data_type: string, read/write

Specifies the directory to store files/sub-directories 'auto-generated' by the tool. This includes report files or other output files with default names that are automatically created by some commands in the current run directory. Default names that have been overridden with a name chosen by the user, are not affected. This allows multiple simultaneous runs from the same run directory to direct output files to different result directories to avoid collisions.

auto_file_prefix

`auto_file_dir` *fileName*

Default: ""

Data_type: string, read/write

Specifies a prefix to be applied to all files 'auto-generated' by the tool. This includes report files or other output files with default names that are automatically created by some commands in the current run directory. So a default name like "cmd.report" will become "<prefix>cmd.report". Default names that have been overridden with a name chosen by the user, are not affected. This allows multiple simultaneous runs from the same run directory to direct default output files to different locations to avoid collisions.

distributed_mmmc_disable_reports_auto_redirection

`distributed_mmmc_disable_reports_auto_redirection {true| false}`

Default: `false`

Data_type: string, read/write

Allows reports to be generated in the specified location.

When set to `true`, the software does not allow the reports for different views to get overwritten. When set to `false`, the reports are generated automatically in the DMMMC view directory structure.

gui_verbose

`gui_verbose {all db none}`

Default: `db`

Data_type: enum {all db none}, read/write

Controls which GUI actions are logged. You can select:

- `all`: Logs all GUI actions.
- `db`: Logs the GUI actions that affect the DB, such as `gui_select`, but does not capture display-only actions such as `gui_dim_foreground`.
- `none`: Does not log any GUI actions. So, the .log and .cmd files are incomplete.

layer_name_no_abbreviation

`layer_name_no_abbreviation {true | false}`

Default: `true`

Data_type: bool, read/write

By default the value of this attribute is true, and most command `-layer <layer>` options require a LEF/OA layer name, or a routing-layer index value (for example, 1 for the first routing layer, 2 for the second) for input. If this attribute is `false`, an older style abbreviation is also allowed (for example, M1 for the first routing layer, M2 for the second routing layer).

Note: This layer abbreviation style is not recommended, and is only allowed for backward compatibility to migrate older scripts. Eventually this attribute will be removed. You should modify your scripts to use LEF/OA layer names, or routing-layer index values for -layer options to avoid problems in the future.

log_verbose_type

```
log_verbose_type {pre post both}
```

Default: `pre`

Data_type: enum {pre post both}, read/write

When verbose logging is enabled, the log can be pre-Tcl expansion, post-Tcl expansion, or both. You can switch between these settings in the middle of a script.

- `pre`: Logs commands pre-Tcl expansion. For example:

  ```
  @file 1: set lef_files $files
  ```

- `post`: In some cases it is helpful to see commands post-Tcl expansion (such as after $files is replaced by its Tcl value). This can result in very long lines, so you should normally set `source_verbose_line_length_limit` if you do this. The example above would look like:

  ```
  @file 1: set lef_files {a.lef b.lef}
  ```

- `both`: You can even log both pre-Tcl and post-Tcl expansion when doing detailed debugging, so you would see each command line repeated like this:

  ```
  @file 1: set lef_files $files
  @file 1: set lef_files {a.lef b.lef}
  ```

Related topics

- `source_verbose` attribute
- `flow_verbose` attribute in the `flow` category
- `set_proc_verbose`

print_full_message_summary

```
print_full_message_summary {true | false}
```

Default: `false`

Data_type: bool, read/write

Controls the format of the message summary. If `true`, prints full message, otherwise prints truncated message.

read_db_directory

```
read_db_directory directory_name
```

Default: ""

Data_type: string, read/write

Specifies the directory of the current loaded database.

read_db_file_check

```
read_db_file_check {true | false}
```

Default: `true`

Data_type: bool, read/write

By default, `read_db` will check all the files inside the saved DB directory. If any file is edited, renamed, or deleted, the tool will give an error. However, read_db will not perform this check if `read_db_file_check` is set to `false`.

read_db_stop_at_design_in_memory

```
read_db_stop_at_design_in_memory {true | false}
```

Default: `true`

Data_type: bool, read/write

By default, tool gives error and stops if you call `read_db` more than once in the same Innovus session unless `read_db_stop_at_design_in_memory` is set to `false`. Using `read_db` twice in one Innovus session is not recommended as settings from the first database may not be cleaned up

properly and can cause problems in the second database.

read_db_tool_name

`read_db_tool_name`

Default: ""

Data_type: string, read/write

Indicates the tool name of the tool that created the restored db on the disk database.

read_db_version

`read_db_version`

Default: ""

Data_type: string, read/write

Indicates the version of the tool that generates an on-disk database.

script_search_path

`script_search_path` *`search_directory_path`*

Default: "."

Data_type: string, read/write

Provides a set of directories for the software to search for files that are sourced using the Tcl `source` command. The software will search in each search path directory for the specified file.

source_echo_filename

`source_echo_filename {true | false}`

Default: `false`

Data_type: bool, read/write

If true, echoes the post-Tcl expansion file name from a source command. This is useful to see “Sourcing a.tcl” that shows the actual ‘a.tcl’ file name rather than a Tcl variable like “source $file”.

set_db_verbose

```
set_db_verbose {true | false}
```

Default: `false`

Data_type: bool, read/write

When true, additional messages are displayed describing every object modified by `set_db` or `reset_db` commands.

Refer to the following example:

```
set_db set_db_verbose true
set_db [get_db hinsts */D*] .dont_touch true
Setting attribute of hinst 'DATA_BUS_MACH_INST': 'dont_touch' = true
Setting attribute of hinst 'DECODE_INST': 'dont_touch' = true
2 true
set_db set_db_verbose false
Setting attribute of root '/': 'set_db_verbose' = false
1 false
set_db [get_db hinsts */D*] .dont_touch true
2 true
```

soft_stack_size_limit

```
soft_stack_size_limit size_limit
```

Default: `10`

Data_type: string, read/write

Specifies the soft stack size limit in mbytes. When the tool is launched, the the soft stack size limit is auto-enlarged to 0.2% RAM. If 0.2%RAM is larger than the hard limit, the tool sets it to the hard limit.

source_verbose

```
source_verbose {true| false}
```

Default: `true`

Data_type: bool, read/write

If true, each command inside a “source file.tcl” script is also logged. For example, the command

executed inside the file will be prefixed with `@file`:

```
@file 1: set_db design_process_node 28
```

All commands have an '@' prefix in the log. If the command is an application command like init_design, or place_design and not a built-in Tcl command like 'set', it would be prefixed with @@. This makes it easy to grep for '@' to see all commands, or '@@' to see just the application commands.

```
@@file 1: init_design
```

When Innovus starts, this attribute is set as `true`. So, commands inside initialization files, such as ./innovus_startup, that are automatically loaded at startup will also be logged. See `man Innovus` for the list of startup files. You can force the startup value to false by setting the following envar before starting Innovus:

```
setenv CDS_STYLUS_SOURCE_VERBOSE 0
```

If you set this attribute to `false`, files are sourced in quiet mode. This is equivalent to source filename -quiet.

source_continue_on_error

```
source_continue_on_error {true| false}
```

Default: `false`

Data_type: bool, read/write

Enable or disables `source` to continue the script on TCL_ERROR. If `true`, the `source` command will ignore Tcl errors, and continue processing the script files. If `false`, a Tcl error will halt the script.

Example

In the following example, there is Tcl File (test.tcl). If `source_continue_on_error` is set to `true`, and incorrect pin is specified for the `get_clock_tree_capacitance` command , the script will continue and source other commands mentioned after `get_clock_tree_capacitance`.

```
@innovus 1> set_db source_continue_on_error true

1 true

@innovus 2> source test.tcl

#@ Begin verbose source test.tcl
```

```
@file(test.tcl) 1: puts "Testing Script for Errors"

Testing Script for Errors


@file(test.tcl) 2: get_clock_tree_capacitance -pin a/b

**ERROR: (IMPTCM-162):  "a/b" does not match any object in design for specified type
"pin port " object in command "get_clock_tree_capacitance".

Usage: get_clock_tree_capacitance [-help] [-edge {rise fall both}] -pin <pin> [ -
delay_corner <delay_corner> ] [ -load | -wire ] [ -delay_type <type> ]


**ERROR: (IMPTCM-4):    The value "a/b" specified for the object type of argument "-
pin" is not a valid object. Review the command specification and remove the argument
or specify a legal value.

**ERROR: (IMPSE-110):   File 'test.tcl' line 2: errors out.

@file(test.tcl) 3: puts "End Testing Script for errors"

End Testing Script for errors

#@ End verbose source test.tcl
```

If `source_continue_on_error` is set to `false`, and incorrect pin is specified for the `get_clock_tree_capacitance` command, the script will end and will not source other commands mentioned after `get_clock_tree_capacitance`.

```
@innovus 3> set_db source_continue_on_error false

1 false

@innovus 4> source test.tcl

#@ Begin verbose source test.tcl

@file(test.tcl) 1: puts "Testing Script for Errors"

Testing Script for Errors

@file(test.tcl) 2: get_clock_tree_capacitance -pin a/b

**ERROR: (IMPTCM-162):  "a/b" does not match any object in design for specified type
"pin port " object in command "get_clock_tree_capacitance".
```

```
Usage: get_clock_tree_capacitance [-help] [-edge {rise fall both}] -pin <pin> [ -
delay_corner <delay_corner> ] [ -load | -wire ] [ -delay_type <type> ]


**ERROR: (IMPTCM-4):    The value "a/b" specified for the object type of argument "-
pin" is not a valid object. Review the command specification and remove the argument
or specify a legal value.

**ERROR: (IMPSE-110):   File 'test.tcl' line 2: errors out.

#@ End verbose source test.tcl
```

log_verbose_style

`log_verbose_style { debug | echo | echo_comments}`

Default: `echo`

Data_type: `enum {debug echo echo_comments}, read/write`

When verbose logging is enabled, the logging style for each command is controllable. It is sometimes helpful to include file name and comments for script debugging and at other times this is too much information. You can switch between settings in the middle of a script.

The values mean:

- `debug`: Will output the file name, line numbers, commands, comments, and empty lines. For example:

  ```
  @file(a.tcl) 1: #comment
  @file(a.tcl) 2: set a $b
  ```

- `echo`: Will output only the line numbers and commands. For example:

  ```
  @file 2: set a $b
  ```

- `echo_comments`: Will output line numbers, commands, comments, and empty lines. For example:

  ```
  @file 1: #comment
  @file 2: set a $b
  ```

Related Topics

- `source_verbose`
- `flow_verbose`
- `set_proc_verbose`

source_verbose_line_length_limit

`source_verbose_line_length_limit` *`value`*

Default: `-1`

Data_type: int, read/write

Limits the length of the verbose source Tcl commands that are logged. -1 means there is no limit. If you use the default pre-Tcl expansion logging, then -1 is generally the best. If you turn on post-Tcl expansion logging (see), a command such as “set a [get_db insts]” will by expanded by Tcl into “set a inst:top/i1 inst:top/i2 …”, and log a list of every inst that will often make the log file too big to be useful. You can limit the length of the longest command line to avoid that. This is different than the Tcl output length limit which only affects the length of the output echoed by Tcl when typing at the prompt.

Related Topics

- `log_verbose_type`
- `tcl_return_display_length_limit`

tcl_error_info

`tcl_error_info {true| false}`

Default: `false`

Data_type: bool, read/write

When set to `true`, the `tcl_error_info` prints the error information if there is an issue in the Tcl script.

tcl_partial_cmd_argument_matching

`tcl_partial_cmd_argument_matching {error | warn | quiet}`

Default: `quiet`

Data_type: enum {quiet warn error}, read/write

Controls argument name and enum name partial matching. An argument name, such as -opt will be accepted for -option if no other argument name begins with -opt.

- `quiet`: Silently allow partial-matching
- `warn`: Gives a warning message but still runs.
- `error`: Disables partial-matching, gives an error, and the command fails.

Related Command

- `get_computed_shapes`

tcl_return_display_length_limit

`tcl_return_display_length_limit`

Default: 10000

Data_type: int, read/write

Limits the number of characters displayed on the xterm window (stdout) and the .log file. The Tcl return value itself is not truncated. A value of `-1` indicates that there is no display limit. Some commands that are likely to have longer Tcl return values are `get_computed_shapes` and `history`. `tcl_return_display_length_limit` is useful to limit the Tcl return output for commands that may have long lists as input. For example, if you type `set a [get_db insts]`, the `set` command will return the value of `$a` to Tcl, and the Tcl interpreter will echo all the insts to stdout and the .log file.

Example

- In the example below, the tcl return values of `get_metric` command options are not truncated. However, the remaining part is truncated to 3 characters.

```
innovus 1> set_db tcl_return_display_length_limit 3
1 3
innovus 2> get_metric *
```

```
flo... truncated to 3 characters (use the tcl_return_display_length_limit
attribute to control the length before truncation).
```

- In the example below, the value of `tcl_return_display_length_limit` is set to `-1`. As a result, there is no display limit.

```
innovus 3> set_db tcl_return_display_length_limit -1
1 -1
innovus 4> get_metric *
flow.cputime 0 flow.cputime.not_child 0 flow.cputime.total 0 flow.cputime.user 0
flow.cputime.user.not_child 0 flow.cputime.user.total 0 flow.realtime 0
flow.realtime.not_child 0 flow.realtime.total 0 metric.version 1
```

ui_precision

`ui_precision` *`number`*

Default: `3`

Data_type: int, read/write

Specifies the number of significant digits to be displayed in the absence of an explicit precision for that type.

ui_precision_capacitance

`ui_precision_capacitance` *`number`*

Default: 3

Data_type: int, read/write

Specifies the number of significant digits to be displayed for quantities of type capacitance.

ui_precision_derating

`ui_precision_derating` *`number`*

Default: 3

Data_type: int, read/write

Specifies the number of significant digits to be displayed for derating factors such as those specified via set_timing_derate constraints or AOCV derating libraries.

ui_precision_power

`ui_precision_power` *number*

Default: 3

Data_type: int, read/write

Specifies the number of significant digits to be displayed for quantities of type power.

ui_precision_sensitivities

`ui_precision_sensitivities` *number*

Default: `3`

Data_type: int, read/write

Specifies the number of significant digits to be displayed for statistical sensitivity values and SOCV sigma values.

ui_precision_timing

`ui_precision_timing` *number*

Default: `3`

Data_type: int, read/write

Specifies the number of significant digits to be displayed for delay type values including cell and net delays, transitions, arrival and required times, and slacks.

write_db_auto_save_user_globals

`write_db_auto_save_user_globals {true| false}`

Default: `false`

Data_type: bool, read/write

If `true`, `write_db` automatically saves the Tcl global variables created by the user. User globals are defined as any “info globals” names that do not exist at startup (such as created by the user executed code).

You can use the `define_variables` command to add or delete specific Tcl variables from the

automatic save list.

write_db_cmd_file_limit

`write_db_cmd_file_limit` *number*

Default: 10

Data_type: int, read/write

The write_db command saves the complete .cmd history across multiple sessions.
The write_db_cmd_file_limit attribute specifies the file length limit (in units of Mb) that is saved in the inn.cmd.gz file inside the save directory. If it is set to 0, no history is saved at all.

uniquify_filename

```
uniquify_filename
[-help]
filename
```

Uniquifies a file.

Parameters

filename	Specifies the name of the file to uniquify. *Data_type*: string, required
-help	Prints a brief description that includes type and default information for each `uniquify_filename` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man uniquify_filename`

update_names

```
update_names
[-help]
[-allowed chars]
[-change_modules hinsts_modules [-local]]
[-design]
[-first_restricted chars]
[-honor_dont_touch]
[-hport]
[-inst]
[-last_restricted chars]
[-log_file string [-append_log]]
[-map {{from to}...}]
[-max_length integer]
[-module]
[-net]
[-no_verbose]
[-nocase]
[-phys_inst]
[-port]
[-prefix string [-name_collision]]
[-random [-random_map_dir out_dir]]
[-regexp]
[-replace_string string]
[-reserved_words string]
[-restricted string]
[-suffix string [-name_collision]]
[-system_verilog]
[-verilog]
[-vhdl]
```

Updates the inst, net, phys_insts, port, module, and design names at each level of hierarchy. The inst and net names are local to only one level of hierarchy, as seen by Verilog inside a module.

If a net or port name is a part of a bus definition, then only the base bus name is updated, and the bus-bits are not modified. So for example, a net named 'busA[0]' specifies that the base bus name is 'busA', which can be changed to 'busB[0]'. However, the '[0]' part will not be changed.

If a net or port name is an escaped scalar name, then all the characters can be modified. So in such a case, an escaped Verilog name like '\busA[0] ' can be changed to 'busA_0_' using various matching options available as shown in the Examples section below.

You can restrict the changes to just one name type using parameters, such as `-inst`, `-net`, `-port`, `-module`, or `-design`. Otherwise, the requested change is applied to all the name types.

By default, there is no restriction on the length of the new names (see `-max_length`).

Each name is checked, and the characters or strings are removed, replaced, or added according to the parameters explained below. Then the final name is checked for collisions with any existing names. If the new name collides with an existing name, a suffix such as _1 and _2 is appended until it is unique.

The old and new names for each modified name are reported.

Note: Names used for SDC timing constraints can only be modified if they are loaded into memory. So you must make sure all the timing views are loaded with `set_analysis_view` before running this command.

Note: There is a limitation that names used inside a CPF `power_intent` file cannot be updated. CPF usage assumes that the CPF file is golden and should not be modified. If you need to change names used inside a CPF file, you need to update the CPF file manually and then read in the new CPF file after running `update_names`. In contrast, an IEEE1801 `power_intent` file is allowed to change during the design flow, so changing any pin, instance or net names with `update_names` will automatically update the IEEE 1801 names. You can write out the new IEEE1801 file with the updated names.

Note: To rename a specific object, use the `rename_obj` command.

Parameters

`-help`	Prints the command usage.
`-allowed` *chars*	Specifies the characters that are allowed in the names. Any characters that are not in the allowed list will be removed in the final name. The minimum specification is 10 characters. To allow all the letters from a to z in capital and lower case letters, you must specify all of them. Use `-regexp` to allow dash "-" as a character range indicator. *Data_type*: string, optional

`-append_log`	Appends the report to `<out_file>` specified with the `-log_file` option. If you omit this option, `<out_file>` is overwritten. You can specify this option only if you have specified the `-log_file` option. *Data_type*: bool, optional
`-change_modules` *hinsts_modules*	Limits the name changes to the objects inside the specified list of modules or modules of the `hinsts`. The changes are applied recursively down the hierarchy below each module, unless the `-local` parameter is specified. The current design is used by default, if this parameter is not used. *Data_type*: `hinst|module|design)+`, optional
`-design`	Changes only the design module name. *Data_type*: bool, optional
`-first_restricted` *chars*	Specifies the characters that cannot be used in the beginning of the name. Those will be removed, including repeated characters. So `-first_restricted {_#%$}` will change _$abc to just abc. *Data_type*: string, optional
`-honor_dont_touch`	Prevents renaming of the objects, for which the `dont_touch` attribute is set.
`-hport`	Changes the port names (e.g. Verilog input, output, or inout names) of the intermediate modules for all hinsts, including bus port names. It also modifies the same implicit net name connected to the port to match the new port name. It does not modify the port names of library cells or the top-level design. *Data_type*: bool, optional
`-inst`	Changes only the Verilog instance names at each level of hierarchy. *Data_type*: bool, optional

`-last_restricted` *chars*	Specifies the characters that cannot be used at the ending of a name. Those will be removed, including repeated chars. *Data_type*: string, optional
`-local`	Turns off the recursive descent, so that changes become local to the module or design. The objects defined in the design/module can be modified (for example, design/module, `port`/`hport`, `inst`, or `net` names), but sub-module names and `hport` names of sub-modules cannot be modified. *Data_type*: bool, optional
`-log_file` *out_file*	Specifies a log file for the output report that shows the old name and the new name for any name changes. *Data_type*: string, optional
`-map {{`*from to*`}...}`	Takes a list of string pairs and maps the first string to the second string in each pair. You need to enclose each pair in braces. To be compatible with Genus usage, double quotes are optionally allowed around the strings. For example, both `-map {{"from" "to"} {"AND" "and"}}` and `-map {{from to} {AND and}}` will result in the same output. *Data_type*: string, optional
`-max_length` *int*	Limits the length of the changed name to the specified number of characters. If the name exceeds the limit, the additional characters will be truncated from the end. If the new name is not unique, the original name is further trimmed back to allow the name with a suffix, such as 1, 2, etc. to make it unique and within the specified length limit. Note, the truncation happens after all other changes. This means the truncated names may now collide with -vhdl, -verilog, or -system_verilog reserved names, or end with a char from the -last_restricted set. If that happens, you will have to rerun the command with the same options and either a longer -max_length or no -max_length. *Data_type*: int, optional

`-module`	Changes all module names other than the design module name. *Data_type*: bool, optional
`-name_collision`	Adds only the `-prefix` or `-suffix` values, if a name change collides with an existing name. This parameter can be used only when either the `-prefix` or `-suffix` parameter is used. *Data_type*: bool, optional
`-net`	Changes the Verilog net names at each level of hierarchy (e.g. Verilog wire names). This will not modify net names that correspond to Verilog port definitions (e.g. Verilog input, output or inout names). If you want to modify those net names, use the -port option. *Data_type*: bool, optional
`-no_verbose`	Suppresses the output generated from `update_names`. *Data_type*: bool, optional
`-nocase`	Changes names that collide if the case is ignored. For example, if both net1 and Net1 exist, one of them will be changed to resolve the collision by appending a suffix, such as _1 or _2. *Data_type*: bool, optional
`-phys_inst`	Changes only those physical inst names (like filler cells) that do not appear normally in the Verilog output. When this parameter is specified, the module-scoping parameters (like `-change_modules`) are ignored, so all the physical inst names are checked. Type: bool, optional

`-port`	Changes the port names (e.g. Verilog input, output, or inout names) of the top-level module for the design, including bus port names. It also modifies the same implicit net name connected to the port to match the new port name. It does not modify the port names of library cells or intermediate modules. *Data_type*: bool, optional
`-prefix` *string*	Adds a prefix to the names that are changed. *Data_type*: string, optional
`-random`	Makes all the modified names random. It obfuscates the design names so that the test cases can be shared without exposing the real netlist names. This parameter cannot be used with any other parameter that affects names. It can be used with `-inst`, `-net`, `-hport`, `-design`, and `-port` to limit the types of names to be modified. *Data_type*: bool, optional

`-random_map_dir` *out_dir*	Specifies the `out_dir` (output directory) for the map-file of the old and new random names to be written for each name type that is modified inside the output directory. This parameter requires the `-random` parameter. It is useful in the cases when there are side files that have names to be mapped and are not supported by `update_names`. The following files are written:
	`inst.map for inst and hinst names`
	`net.map for net and hnet names`
	`pin.map for hport and pin names`
	`port.map for top-level port names`
	`module.map for module names`
	The format is *old_name new_name* on each line for every name. For example, `inst.map` may look like:
	`blockA/blockB/regC i1/i2/i3`
	`blockA/regD i1/i3`
	…
	Data_type: string, optional

`-regexp`	Allows specifying character ranges with a dash "-" in the `-allowed` option. This is similar to regexp single character matching usage. For example, to allow all alphanumerical characters, along with _ and $ as legal chars, use: -allowed {a-zA-Z0-9_$} *Data_type*: bool, optional
`-replace_string` *string*	Specifies the replacement string for `-restricted` or `-reserved_words` strings. The default value for this option is "_". *Data_type*: string, optional
`-reserved_words` *string*	Specifies a list of words to be avoided (reserved words), like 'begin end'. If a complete name exactly matches one of the reserved words (not just a sub-string of the name), then by default, the name will be changed with the addition of an `_1`, `_2`, etc. suffix. If you also specify the `-replace_string` *rep_str* parameter, then the reserved words will be replaced by *rep_str* instead, and the `_1`, `_2`, etc. will be appended, if there are any collisions. *Data_type*: string, optional
`-restricted` *string*	Specifies a list of strings that cannot be used inside a name. Each string is replaced with the `-replace_string` string. For example, `-restricted {ab bc}` will replace all the `ab` and `bc` strings inside the names. *Data_type*: string, optional
`-suffix` *string*	Adds a suffix to the names that are changed. **Note:** If the length of `-suffix` exceeds the `-max_length` value, the suffix is truncated.
`-system_verilog`	Replaces SystemVerilog reserved words. If a complete name exactly matches one of the reserved words (not just a sub-string of the name), then the name will be changed. *Data_type*: bool, optional

`-verilog`	Replaces Verilog reserved words. If a complete name exactly matches with one of the SystemVerilog reserved words (not only a sub-string of the name), then the name will be changed. *Data_type*: bool, optional
`-vhdl`	Fixes names to meet VHDL identifier requirements. It replaces VHDL reserved words. If a complete name exactly matches one of the VHDL reserved words (not only a sub-string of the name), then the name will be changed. It also implicitly uses –first_restricted “_0123456789” to remove any leading _ or digit chars, and –last_restricted “_” to remove any trailing _ chars, so you cannot use –vhdl together with –first_restricted or –last_restricted. Finally, this option will remove any repeated _ chars in the middle of any name, and replace it with a single _ char because VHDL does not allow repeated _ chars. *Data_type*: bool, optional

Examples

- The following command replaces the `p1` inst name with the `prefix_p1` prefix without updating all the `p1` sub-strings (for example, without changing `insts` named `p12` or `op1`):
  ```
  update_names -inst -reserved_words {p1} -replace_string {prefix_p1}
  ```
- The following example replaces sub-string "no" with "NO", and all brackets "[" and "]" with underscore "_" in all net names:
  ```
  innovus 1> update_names -net -map {{no NO} {[ _} {] _}}
  ```
  Optional double-quotes are ignored, so this has the same result:
  ```
  innovus 1> update_names -net -map {{"no" "NO"} {"[" "_"} {"]" "_"}}
  ```
  So, a Verilog net name that requires escaping like:
  `\test[0]nox` would change to `test_0_NOx`
  Note, neither the Verilog or DEF name requires escaping anymore, so no \ would be written to Verilog or DEF for these names. A “real” bus name like
  `no[0]` would just be changed to `NO[0]`
  because the `[0]` are bus-bit chars and values that are never changed.

- The following example replaces any sub-strings that match ab, bc, or ca with "@" in all object names:
  ```
  innovus 1> update_names -restricted "ab bc ca" -replace_string "@"
  ```
- The following command changes the brackets "[" and "]" in all names to "_". It does not change real bus-bit chars for nets.
  ```
  innovus 1> update_names -restricted {[ ]} -replace_string "_"
  ```
  Note: You should use {} to protect the [] from being interpreted by Tcl.
- The following example specifies the maximum length of all module names to be 12 characters.
  ```
  innovus 1> update_names -max_length 12 -module
  ```
- The following example fixes case-insensitive collisions. Suppose you have two insts named inst73 and INST73. This command renames one of the insts by appending _<num> to one of the names for avoiding a naming conflict, when casing is ignored.
  ```
  innovus 1> update_names -inst -nocase
  ```
- The following example allows all capital and lower case letters, numbers, underscores, backslash, and brackets. Any other character will be removed.
  ```
  innovus 1> update_names -allowed
  {ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789_\[]}
  ```
  If you want to use character ranges, you can use `-regexp`:
  ```
  innovus 1> update_names -regexp -allowed {a-zA-Z0-9_\[]}
  ```
- If you want to change names that collide with reserved Verilog words, then use the following command:
  ```
  innovus 1> update_names -verilog
  ```
  Names that exactly match a Verilog reserved words, like "input" will be replaced with "input_1". This does not replace sub-strings in names that match "input", so "my_input" would not be modified.

Related Topics

`rename_obj`

view_log

```
view_log
[-help]
[file_name logFileName]
```

Opens up a log file in a separate console window for viewing within the Innovus.

The log file viewer within the Innovus opens the current log file by default, and updates it in real time. If you specify a log file name, the software opens up the specified log file.

The log file displayed contains `[+]` and `[-]` markers that expand and collapse command information, so that you can control how much detail you want to read. Log messages are color coded for easier identification: blue messages are warnings, red messages are errors.

You can access documentation for commands in the log file that are underlined. Click on the command in the file, and the software opens an HTML window displaying the *Innovus Text Command Reference* documentation for that command.

You also can access the log file viewer from the *Tools Menu- Log Viewer*.

Note: You can specify `view_log` more than once to view multiple log files in separate console windows simultaneously. However, you cannot open multiple versions of the current log file.

Parameters

`file_name` *`logFileName`*	Specifies the name of the log file to open. *Data_type*: string, optional
`-help`	Outputs a brief description that includes type and default information for each `view_log` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man view_log`.

Related Information

- Tools Menu- *Log Viewer*.

gui_get_legend

```
gui_get_legend
[-help]
-type {inst_density pin_density metal_density route_congestion}
[-display_range | -value_range | -show_legend_on_layout | -legend_location | -
enable_min_value | -enable_max_value]
```

Gets the current legend setting.

You can use the `gui_set_legend` and `gui_get_legend` commands to get the snapshot of different density for analysis.

Parameters

`-help`	Prints a brief description of the command. For a detailed description of the command, use the `man` command: `man gui_get_legend`
`-display_range`	Gets the display range
`-enable_min_value`	Gets the state of the min value of display range.
`-enable_max_value`	Gets the state of the max value of display range.
`-legend_location`	Gets the location of legend.
`-show_legend_on_layout`	Gets the state of legend on layout.
`-type {inst_density pin_density metal_density route_congestion}`	Gets the setting for the specified legend type.
`-value_range`	Gets the value range.

Example

The following command sets the legend for the inst_density legend type:

```
> gui_set_legend -type inst_density -display_range {1 0.9} -show_legend_on_layout true
```

The following command gets the current legend setting for the `inst_density` legend type.

```
> gui_get_legend -type inst_density
{display_range {0.9 0.9}} {legend_location ne} {show_legend_on_layout true}
{value_range {0.75 1}}
```

Related Information

- `gui_set_legend`

gui_set_legend

```
gui_set_legend
[-help]
[-display_range {x y}]
[-enable_max_value {true false}]
[-enable_min_value {true false}]
[-legend_location {ne nw se sw}]
[-reset -display_range|-value_range|-show_legend_on_layout|-enable_min_value|-
enable_max_value]
[-show_legend_on_layout {true false}]
-type {inst_density pin_density metal_density route_congestion}
[-value_range {x y}]
```

Specifies the legend setting.

You can use the `gui_set_legend` and `gui_get_legend` commands to get the snapshot of different density for analysis.

Parameters

`-help`	Prints a brief description of the command. For a detailed description of the command, use the `man` command: `man gui_set_legend`
`-display_range {x y}`	Specifies the display range in the {min max} format
`-enable_max_value {true false}`	Enables the max value of the display range.
`-enable_min_value {true false}`	Enables the min value of the display range.
`-legend_location {ne nw se sw}`	Specifies the location of legend.
`-reset` *`-display_range|-value_range|-show_legend_on_layout|-enable_min_value|-enable_max_value`*	Resets the settings of the specified option, `-reset -display_range|-value_range|-show_legend_on_layout |-enable_min_value|-enable_max_value`. You can reset all options by using `-reset all`.
`-show_legend_on_layout {true false}`	Specifies to show the legend on layout.
`-type {inst_density pin_density metal_density route_congestion}`	Specifies the setting for the specified legend type.
`-value_range {x y}`	Specifies the value range in the {min max} format.

Example

The following command sets the legend for the inst_density legend type:

```
> gui_set_legend -type inst_density -display_range {1 0.9} -show_legend_on_layout true
```

The following command gets the current legend setting for the `inst_density` legend type.

```
> gui_get_legend -type inst_density
{display_range {0.9 0.9}} {legend_location ne} {show_legend_on_layout true}
```

```
{value_range {0.75 1}}
```

Related Information

- gui_get_legend

read_taf

```
read_taf
[-help]
[filename [-quiet]
```

Applies additional attribute data from Genus for the user-managed exchanges. It is not needed for managed exchanges after iSpatial (syn_opt -spatial) from Genus to Innovus or when using common-DB between Genus and Innovus. The 'taf' format is a private format between Genus and Innovus.

Parameters

`-help`	Prints a brief description that includes type and default information for each `read_taf` parameter.
filename	Specifies the name of file to read. *Data_type*: string, required
`-quiet`	When `-quite` is specified, the tool does not give a summary. *Data_type*: bool, optional

report_disk

```
report_disk
[-help]
[-dir directory_name]
```

Reports the disk metrics of the current working directory and the /tmp directory. The output of this command allows you to investigate Mb/sec throughput of the disk and disk I/O requirements.

This command also reports the used and free disk space for the directory specified by the `-dir` parameter (default is /tmp).

Parameters

`-help`	Outputs a brief description that includes type and default information for each `report_disk` parameter. For a detailed description of the command and all of its parameters, use the man command: `man report_disk`.
`-dir` *`directory_name`*	Specifies the name of the directory in which disk metric information is written. The default directory is `/tmp`.

Examples

- The following is a sample output for `report_disk`:

```
>report_disk

DISK_INFO : 206.66 Mb/S   381G total   1.1G used   359G free   /env(TMPDIR)

{206.66 381G 1.1G 359G}
```

- The following command shows the usage of `report_disk -dir`:

```
>report_disk -dir .

DISK_INFO : 98.51 Mb/S   9.6T total   9.3T used   395G free   .

{98.51 9.6T 9.3T 395G}
```

time_info

```
time_info
[-help]
[stage_name]
[-current]
[-report]
[-stamp]
[-table metric_table_id]
```

Reports and tracks runtime and memory performance.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `time_info` parameter. For a detailed description of the command and all of its parameters, use the man command: `man time_info`.
stage_name	Specifies the name of the stage.
`-current`	Reports the current stage.
`-report`	Reports all stages up to this point
`-stamp`	Creates a time stamp.
`-table` *metric_table_id*	Specifies metrics table ID to support multiple tables. *Default*: 'default'

Examples

- The following command shows the usage of the `-current` parameter:

```
[DEV]innovus 4> time_info -current
=========================================================
CPU Runtime: 0:00:52.5
Wall Runtime: 0:03:41
Memory Usage: 1387.4 MBytes
Peak Memory Usage: 1505.3 MBytes
Current Time: 12:17:13 (Jul07)
=========================================================
```

- The following command shows the suage of -report:

```
@[DEV]innovus 6> time_info -report
Total Wall | Stage Wall | Wall % | Date - Time     | Memory   | Stage
----------------+----------------+------------+-------------------+------------+--
--------------------
0:00:00    | 0:00:00    | 0.0%   | 8:27:08 (Jul07) | 824.3 MB | init
----------------+----------------+------------+-------------------+------------+--
--------------------
Number of threads: 1 * 1 (id: default)
```

14

GUI Commands

- clear_all_rulers
- create_gui_shape
- create_gui_text
- create_ruler
- delete_workspace
- edit_copy
- gui_add_marker
- gui_add_ui
- gui_attach_to_cursor
- gui_browse_db
- gui_clear_highlight
- gui_clear_hinst_colors
- gui_clear_trace_flightline
- gui_close_cell_view
- gui_create_floorplan_snapshot
- gui_create_user_bundle_net
- gui_delete_objs
- gui_delete_ui
- gui_delete_user_bundle_net
- gui_deselect
- gui_dim_foreground
- gui_find_ui

- gui_fit
- gui_get_box
- gui_get_coord
- gui_get_coords
- gui_get_ui
- gui_get_visible_bumps
- gui_get_visible_net_groups
- gui_get_visible_nets
- gui_group_hinst
- gui_hide
- gui_highlight
- gui_highlight_pin
- gui_highlight_pin_connection
- gui_list_marker
- gui_open_cell_view
- gui_open_schematic
- gui_pan
- gui_pan_center
- gui_pan_page
- gui_redo
- gui_redraw
- gui_register_edit_callback
- gui_remove_marker
- gui_report_trace_flightline
- gui_schematic_load_view
- gui_schematic_save_view
- gui_select

- gui_select_highlighted
- gui_set_obj_color
- gui_set_power_rail_display
- gui_set_power_rail_layers_nets
- gui_set_select_hinst_color
- gui_set_tool
- gui_set_ui
- gui_set_visible_bumps
- gui_set_visible_net_groups
- gui_set_visible_nets
- gui_show
- gui_show_edge_number
- gui_show_floorplan_snapshot
- gui_show_user_bundle_nets
- gui_trace_flightline
- gui_undo
- gui_ungroup_hinst
- gui_unregister_edit_callback
- gui_view_box
- gui_write_flightline
- gui_write_flow_gifs
- gui_write_picture
- gui_zoom
- read_power_rail_results
- read_workspace
- report_gui_edit_callback
- report_power_rail_results

- set_default_workspace
- view_last
- view_next
- write_color_preference
- write_hinst_colors
- write_to_gif
- write_workspace

clear_all_rulers

```
clear_all_rulers -help
```

Removes all ruler lines from the main window. You can also clear the ruler lines with the *Clear All Rulers* icon on the toolbar in the main window.

Parameters

`-help`	Prints a brief description that includes type and default information for each `clear_all_rulers` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man clear_all_rulers`

Related Information

- `create_ruler`
- The Main Window chapter of *Menu Reference*
  - *Toolbar Widgets*
- Tools Menu chapter of *Menu Reference*
  - *Create Ruler*
  - *Clear All Rulers*

create_gui_shape

```
create_gui_shape
[-help]
[-layer layerName]
[-name guiShapeName]
[-width width]
{-rect {x1 y1 x2 y2} | -line {x1 y1 x2 y2 ...} | -polygon {x1 y1 x2 y2 ...}}
[ -line {x1 y1 x2 y2 ...} [-arrow ] ]
```

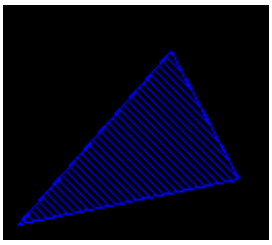
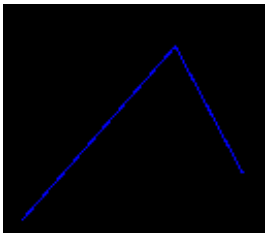
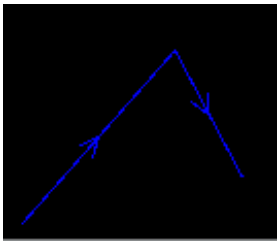
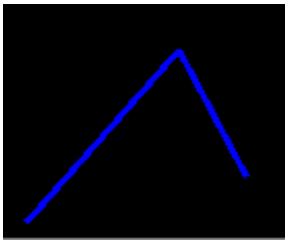
Adds a shape to the default or specified custom layer. You can use the `create_gui_shape` command at any point in the design flow. The shapes added by the command annotate comment shapes and do not show up in a DEF output.

Note: In OpenAccess flows, the shapes added with the `create_gui_shape` command are stored in the OpenAccess cellview. However, the shapes are stored in a private structure that is accessible only to Innovus. To create shapes that are visible in Virtuoso as well, use the `create_shape` command instead.

Parameters

`-help`	Prints a brief description that includes type and default information for each `create_gui_shape` parameter. For a detailed description of the command and all of its parameters, use the man command: `man create_gui_shape`
`-arrow`	Draws an arrow in the middle of the line segment to indicate the start to end direction. This parameter can only be used with `-line`. *Data_type*: bool, required
`-layer` *layerName*	Specifies the GUI layer name to use or create for the shape. If the specified layer does not exist, it is created automatically. If a layer is not specified, the tool automatically creates a layer with the name `defaultLayer` and adds the shape to it. *Data_type*: string, optional
`-line {`*x1 y1 x2 y2 ...*`}`	Creates a line, with *x1 y1 x2 y2 ...* specifying the coordinates for the line segments. The values are specified in microns. *Data_type*: (point)+, optional
`-name` *guiShapeName*	Specifies the name of the GUI shape. *Data_type*: string, optional
`-polygon {`*x1 y1 x2 y2 ...*`}`	Creates a polygon shape, with *x1 y1 x2 y2 ...* specifying the coordinates for the polygon segments. The values are specified in microns. *Data_type*: (point)+, optional
`-rect {`*x1 y1 x2 y2*`}`	Specifies the coordinates of the lower-left and upper-right points of the rectangle. The values are specified in microns. *Data_type*: rect, optional
`-width` *width*	Specifies the width of the shape border in pixels. The minimum (default) value is `1` and the maximum is `7`. *Data_type*: int, optional

Examples

- The following command adds a triangular shape in layer `1`:
  ```
  create_gui_shape -layer 1 -polygon {95.841 -645 96.592 -644.147 96.926 -644.777}
  ```
  
- The following command adds a line shape with two segments:
  ```
  create_gui_shape -layer 1 -line {95.841 -645 96.592 -644.147 96.926 -644.777}
  ```
  
- The following command adds a line shape with two segments and direction arrows:
  ```
  create_gui_shape -layer 1 -line {95.841 -645 96.592 -644.147 96.926 -644.777} -
  arrow
  ```
  
- The following command adds a line shape of 3-pixel width:
  ```
  create_gui_shape -layer 1 -line {95.841 -645 96.592 -644.147 96.926 -644.777} -
  width 3
  ```
  

Related Information

- create_gui_text
- create_shape
- create_text
- The Main Window chapter of *Menu Reference*
  - *Wire Edit Toolbar*

create_gui_text

```
create_gui_text
[-help]
-label text
-layer layerName
{-box {x1 y1 x2 y2} | {{-height height | -fixed_height fixHeight} -pt {x y} [-
orient direction]}}
[-box {x1 y1 x2 y2} [-no_bbox]]
```

Adds text (annotation) to the specified custom layer. You can use the `create_gui_text` command at any point in the design flow. The text added by the command annotates comment text and does not show up in a DEF output.

Note: In OpenAccess flows, any text added with the `create_gui_text` command is stored in the OpenAccess cellview. However, the text is stored in a private structure that is accessible only to Innovus. To create text that is visible in Virtuoso as well, use the create_text command instead.

Parameters

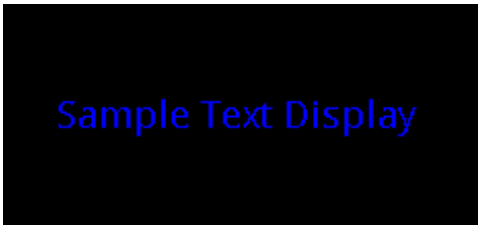
`-help`	Prints a brief description that includes type and default information for each `create_gui_text` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man create_gui_text`
`-box {x1 y1 x2 y2}`	Specifies the coordinates of the lower-left and upper-right points of the rectangle in which text will be added. The coordinate values are specified in microns. When this parameter is used, the tool figures out the correct size and orientation to be used to fit the text into the box. The text is center justified in the box. Use this parameter to control the text orientation and size. *Data_type*: rect, optional
`-fixed_height fixHeight`	Specifies the height in pixels. `fixHeight` should be an integer value. *Data_type*: int, optional

`-height` *height*	Specifies the height of the text in microns. *Data_type*: float, optional
`-label` *text*	Specifies the text that you want to add. *Data_type*: string, required
`-layer` *layerName*	Specifies the GUI layer name to use or create for the text. If the specified layer does not exist, it will be automatically created. *Data_type*: string, required
`-no_bbox`	Specifies that bounding box should not be drawn when text is displayed at the specified box coordinates. This parameter can be used only when `-box` is specified. *Data_type*: bool, optional
`-orient` *direction*	Specifies the orientation of the text. Possible values are: `mx mx90 my my90 r0 r90 r180 r270` This option cannot be used with `-box`.
`-pt {`*x y*`}`	Specifies the origin coordinate of the text in microns. *Data_type*: point, optional

Examples

- The following command adds the text label `Sample Text Display` with size of 18 pixels on layer `1` at the specified point coordinates:

  ```
  create_gui_text -layer 1 -label "Sample Text Display" -fixed_height 18 -pt {0 250}
  ```

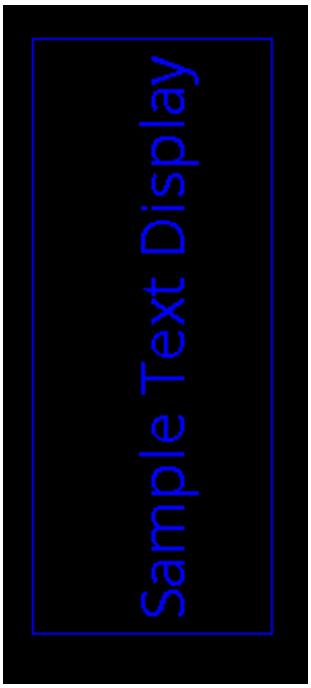
  

- The following command adds the text label `Sample Text Display` on layer `1` in a box with the specified coordinates. As the length of the box is more than the width, the text is displayed horizontally:

```
create_gui_text -layer 1 -box {0 250 500 450} -label "Sample Text Display"
```

- The following command adds the text label `Sample Text Display` on layer `1` in a box with the specified coordinates. As the height of the box is more than the width, the text is displayed vertically:

  ```
  create_gui_text -layer 1 -box {0 250 200 750} -label "Sample Text Display"
  ```



- The following command adds the text label `Sample Text` Display on layer `1` at the specified box coordinates. However, the bounding box is not displayed:

  ```
  create_gui_text -layer 1 -box {0 250 500 450} -no_bbox -label "Sample Text
  Display"
  ```

Related Information

- `create_gui_shape`
- `create_shape`
- `create_text`
- The Main Window chapter of *Menu Reference*
  - *Wire Edit Toolbar*

create_ruler

```
create_ruler
[-help] {{-center {x y} -radius value} | -coords {x1 y1 x2 y2 ...}}
```

Adds a ruler. For an orthogonal ruler, use `-coordinates` to specify ruler segments. For a circular ruler, use `-center` and `-radius`.

Parameters

`-help`	Prints a brief description that includes type and default information for each `create_ruler` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man create_ruler`
`-center {x y}`	Specifies the coordinates for the center of a circular ruler. *Data_type*: point, optional
`-coords {x1 y1 x2 y2 ...}`	Specifies the beginning and ending coordinates for the ruler line segments. The values are in microns. *Data_type*: (point)+, optional
`-radius value`	Specifies the radius of a circular ruler. *Data_type*: double, optional

Example

- The following command creates a circular ruler of radius 500 with the center at `{0 0}`:

  ```
  create_ruler -center {0 0} -radius 500
  ```

Related Information

- `clear_all_rulers`
- The Main Window chapter of *Menu Reference*

- *Toolbar Widgets*

- Tools Menu chapter of *Menu Reference*
  - *Create Ruler*
  - *Clear All Rulers*

delete_workspace

```
delete_workspace
[-help]
[-dir directory]
-name workspaceName
```

Deletes the specified workspace. Use this command at any stage in the design flow to remove a workspace that you do not need any longer.

Parameters

`-help`	Prints a brief description that includes type and default information for each `delete_workspace` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man delete_workspace`
`-dir` *`directory`*	Specifies the directory from which the workspace is to be deleted. The default search order to access directories for deleting workspaces is as below: • Current working directory • User home directory *Data_type*: string, optional
`-name` *`workspaceName`*	Specifies the name of the workspace to be deleted. *Data_type*: string, required

Example

The following example deletes the `Test` workspace from the current working directory.

```
delete_workspace -name Test
```

Related Information

- `read_workspace`
- `write_workspace`
- `set_default_workspace`
- Windows Menu chapter of the *Menu Reference*
    - *Workspaces*

edit_copy

```
edit_copy
[-help]
[-orientation {r0 r90 r180 r270 mx mx90 my my90}]
[-net net_name | -net_group string| -keep_net_name]
[-rotate_point {x y} | -times integer]
[-rotate_point {x y} | [dx dy ]]
```

Copies selected or specified objects to the specified location. In the current release, the command supports copying of regular wires and vias, special wires and vias, as well as patch wires. When `set_db edit_wire_snap_objects_to_track regular` (or `special` or `patch`) is specified before using `edit_copy`, copied wires of that type are snapped to track.

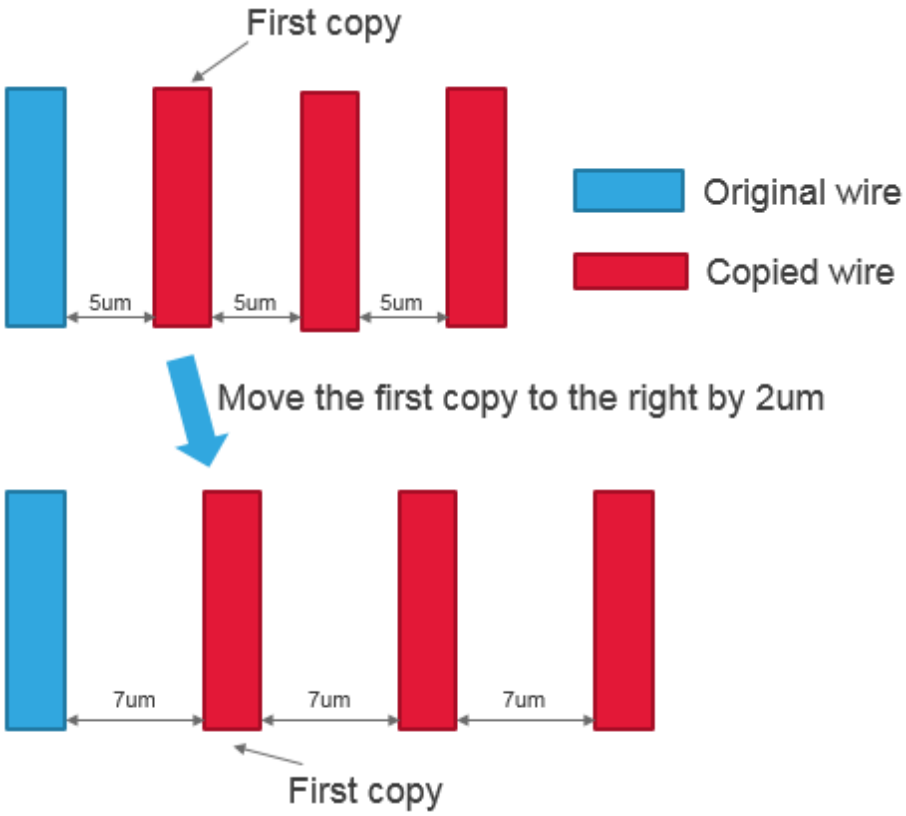
Note: Use the `copy_obj` and `paste_obj` commands to copy placement and routing blockages.

Parameters

`-help`	Prints a brief description that includes type and default information for each `edit_copy` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man edit_copy`
dx	Specifies the dx distance to move the object copy in microns. *Data_type*: float, required
dy	Specifies the dy distance to move the object copy in microns. *Data_type*: float, required
`-keep_net_name`	Specifies that the net name from the original object should be retained after copying. *Data_type*: bool, optional

`-net` *net_name*	Specifies net to which the copied object should be assigned. If neither `-net` nor `-keep_net_name` is specified and: • The copied wire connects to a single wire, the tool assigns the single wire's net to the copied wire. • The copied wire connects to multiple wires on the same net, the tool assigns the net to the copied wire. • The copied wire connects to multiple wires on different nets, the tool assigns `_NULL` to the copied wire and displays the following message: `**WARN: (IMPSPR-304): One target route connects to more than one routes on different nets, net for the target route might not be correctly assigned. Undo and try again to pick a proper location.` *Data_type*: (net), optional
`-net_group` *string*	Assigns the specified net group name to the new copy of the bus guide. You can specify only one net group name with the `-net_group` parameter. If the net group name you specify does not exist, the tool does not copy the bus guide and issues an error message. If you copy a bus guide without the `-net_group` parameter, the new bus guide retains the net group name of the original bus guide. Note that the `-net_group` parameter cannot be used for wires or vias. If you use it for copying a wire or via, the tool ignores the setting but continues the copy operation with a warning. Similarly, the `-net` and `-keep_net_name` parameters do not work for bus guides. If you copy a bus guide with `-net` or `-keep_net_name`, the tool ignores the settings, issues a warning, and copies the bus guide with the original net group name. *Data_type*: string, optional
`-orientation {r0 r90 r180 r270 mx mx90 my my90}`	
	Specifies the orientation of object copy after pasting. Use this parameter to rotate the selected wire or via during copy/paste. For values `r0`, `r90`, `r180`, and `r270`, the wire or via is rotated by specified degrees around the lower left point. For values `mx`, `mx90`, `my`, and `my90`, the bounding box is mirrored. *Data_type*: enum, optional

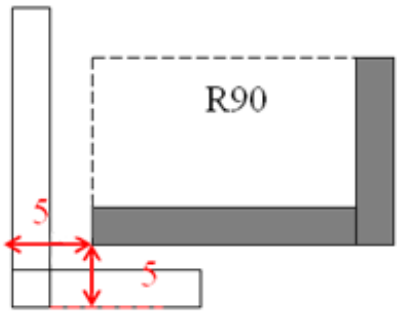
`-rotate_point {x y}`	
	Rotates the copied wire(s) around the specified point `{x y}`, when specified along with the `-orientation` parameter. If the `-rotate_point` parameter is not specified, the copied wires are rotated by default around the lower-left corner of the bounding box of the original wires. If you specify the `-rotate_point` parameter without the `-orientation` parameter, the tool copies the object at the same location. This is the same as specifying the `-orientation {R0}` option. **Note**: The `-rotate_point` parameter cannot be used with the `-times` parameter.

`-times` *integer*	Specifies the number of times the object is to be copied. For example, if you specify 3, the selected wire is copied thrice. The distance between different copies of the wire is equal to the distance between the original wire and the first copy. If you move the first copy, the other copies are moved by the same distance:  **Note**: If DRC checking is turned on with `setEditMode -drc_on true`, all copied wires obey the DRC rules. *Data_type*: int, optional

Example

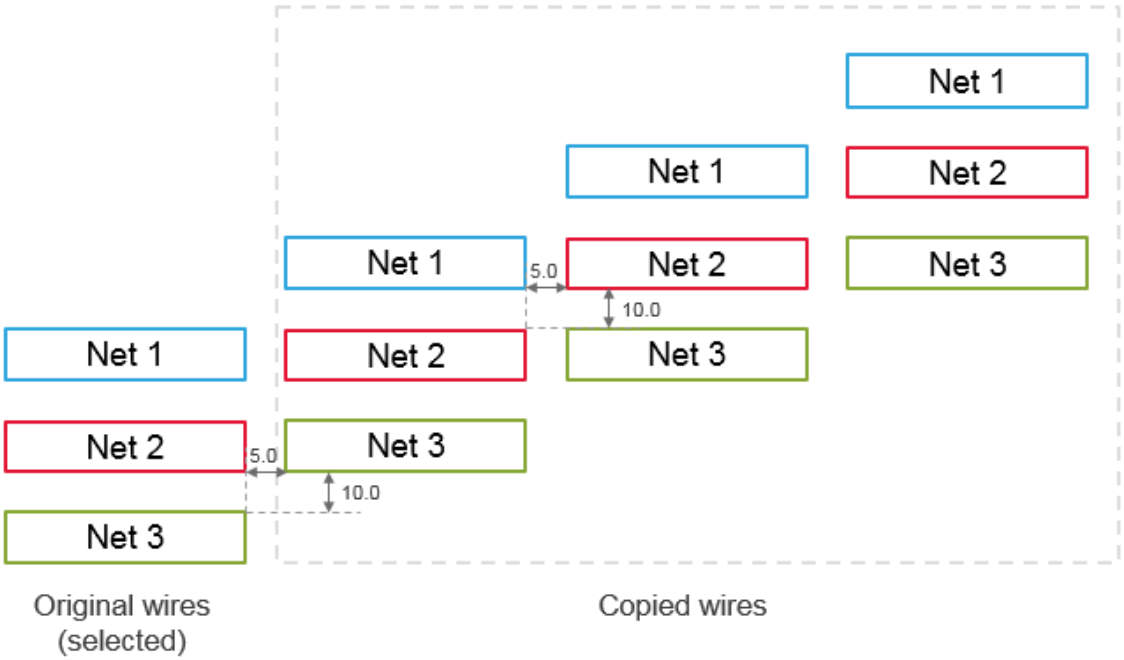
- The following command copies the selected wire 5 microns up and to the right of the original and rotates the wire by 90 degrees around the lower left corner:

```
edit_copy 5 5 -orientation r90
```



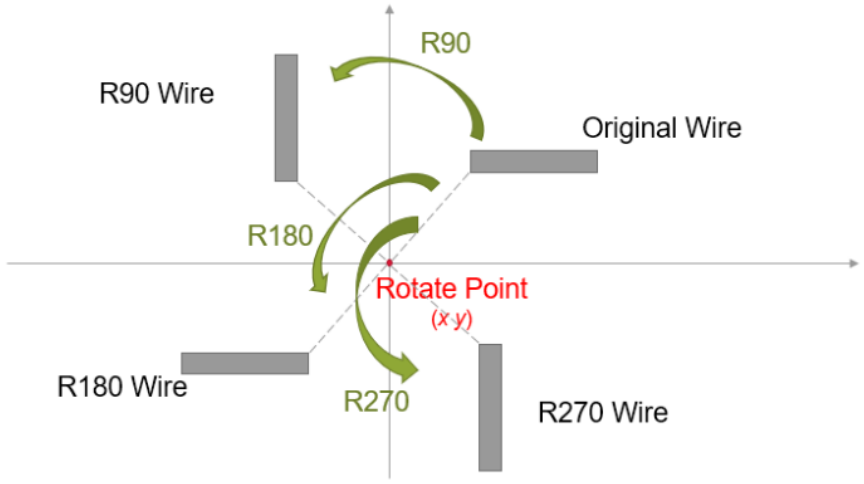
- The following command copies the selected wires 3 times and the distance between each copy set is as specified:

```
edit_copy 5.0 10.0 -times 3 -keep_net_name
```



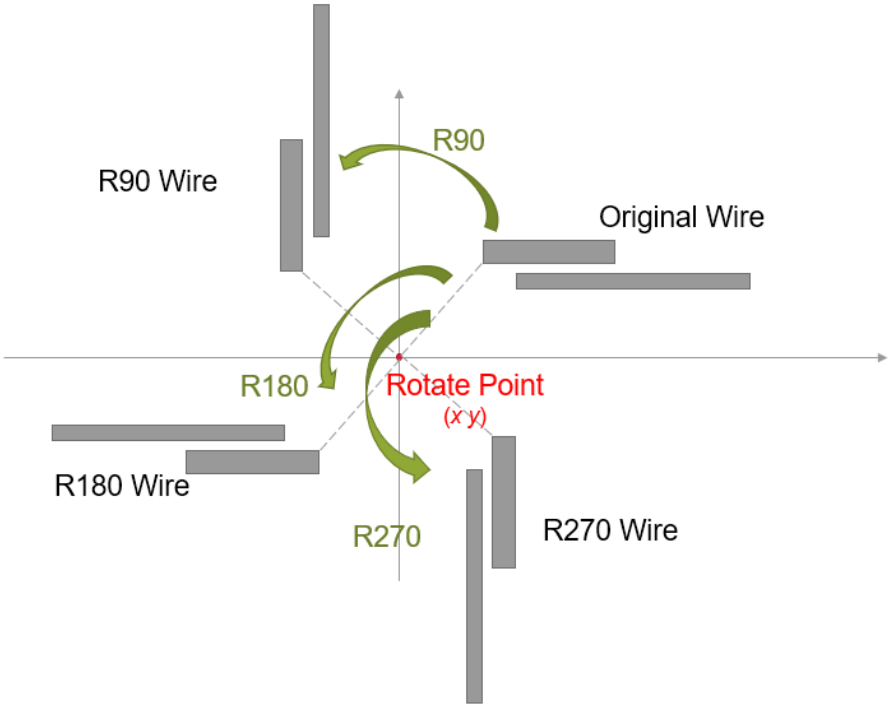
- The following commands rotate the new copy of the selected wire by 90, 180, and 270 degrees, respectively, around the specified rotation point, *x y*:

```
edit_copy -orientation r90 -rotate_point x y
edit_copy -orientation r180 -rotate_point x y
edit_copy -orientation r270 -rotate_point x y
```



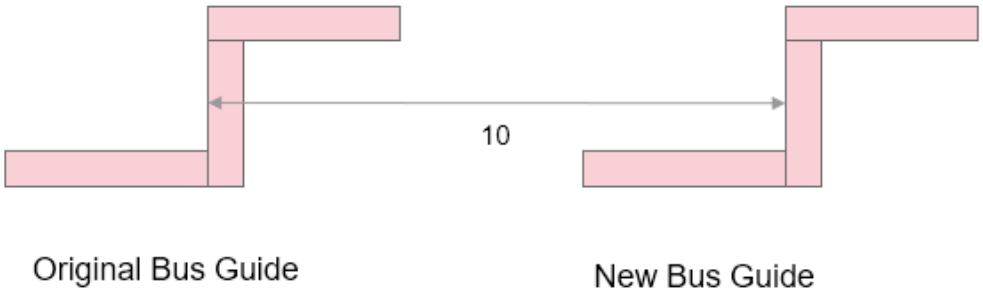
- The following commands rotate the new copy of the selected group of wires by 90, 180, and 270 degrees, respectively, around the specified rotation point, *x y*:

```
edit_copy -orientation r90 -rotate_point x y
edit_copy -orientation r180 -rotate_point x y
edit_copy -orientation r270 -rotate_point x y
```



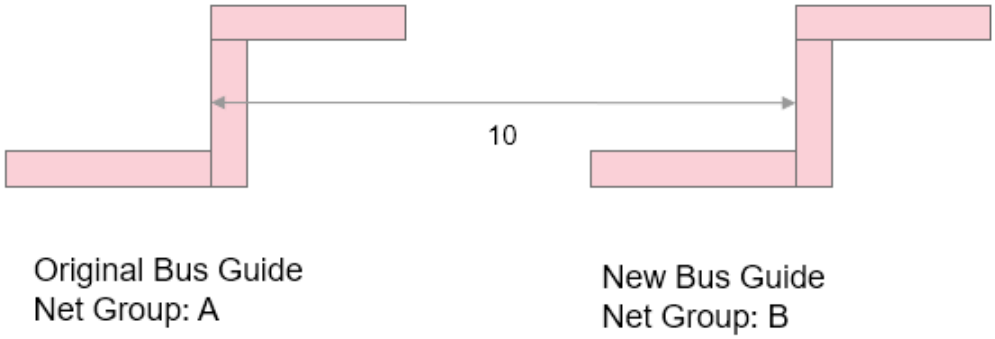
- The following command copies the selected bus guide 10 microns to the right of the original:

```
edit_copy 10 0
```



- The following command copies the selected bus guide 10 microns to the right of the original and assigns the net group `B` to the copy:

```
edit_copy 10 0 -net_group B
```



Related Information

- `copy_obj`
- `edit_duplicate_routes`
- `edit_move_routes`
- `paste_obj`
- Attributes in the `edit_wire` category
- Edit Menu chapter of the *Menu Reference*
  - *Copy* command
- Editing Wires chapter of the *User Guide*
  - *Copying Wires*

gui_add_marker

```
gui_add_marker
[-help]
-color <string>
-name <string>
-point {x y}
-type {x tick star}
```

Places a marker to indicate location on the design layout.

Parameters

`-help`	Outputs the command usage and a brief description about the command parameters.
`-color <string>`	Specifies the color of the marker. You can specify 16 colors for the marker. These are: `red blue green yellow magenta cyan pink orange brown purple violet teal olive gold maroon wheat` *Data_type*: string, required
`-name <string>`	Specifies the name of the marker. *Data_type*: string, required
`-point {x y}`	Specifies the location `{x y}` at which the marker will be placed on the design layout. *Data_type*: point, required
`-type {x tick star}`	Specifies the shape of the marker. There are three shapes for the markers: `x`, `tick`, and `star`. *Data_type*: enum, required

Example

- The following command shows that a X type marker of color RED has been placed at the location {100 100}.
  ```
  gui_add_marker -name aa -color red -point {100 100} -type x
  ```

Related Information

- gui_list_marker
- gui_remove_marker

gui_add_ui

```
gui_add_ui
[-help]
name
[-before name]
[-checked {true | false}]
[-command string]
[-disabled {true | false}]
[-enable_variable string]
[-foreground string]
[-icon filename]
[-in name]
[-label string]
[-movable {true | false}]
[-newline {true | false}]
[-shortcut string]
[-tooltip string]
-type {menu | toolbar | submenu | command | check | radio | separator | toolbutton}
[-underline integer]
[-value string ]
[-variable string]
[-visible_variable string]
```

Adds a new interface element with the specified name of the specified type. Use this command to customize the Innovus graphical interface by adding menus, toolbars, status bar, and other interface elements, as required.

Parameters

`-help`	Prints a brief description that includes type and default information for each `gui_add_ui` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man gui_add_ui`

name	Specifies as a string the internal name of the interface element you want to add. **Note:** Choose the internal name of the element carefully as this is what is used internally by the software to identify the element for any further manipulation. *Data_type*: string, required
-before *name*	When specified, adds the new element before the element identified by *name*. *Data_type*: string, optional
-checked {true | false}	Specifies whether or not the check box or radio button you are adding should be checked. **Note:** You can use the `-checked` parameter only if you are adding elements of type `check` or `radio`. *Default*: `false` *Data_type*: bool, optional
-command *string*	Specifies the Tcl command to be executed when the element you are adding is invoked. **Note:** You can use the `-command` parameter only if you are adding elements of type `command, check, radio` or `toolbutton`. *Data_type*: string, optional
-disabled {true | false}	Specifies the state of the interface element you are adding. **Note:** You can use the `-disabled` parameter only if you are adding elements of type `menu, submenu, command, check, radio` or `toolbutton`. The parameter is NOT applicable for elements of type `toolbar` and `separator`. *Default*: `false` *Data_type*: bool, optional
-enable_variable *string*	Represents the Tcl global variable for Enable status. *Data_type*: string, optional

`-foreground` *`string`*	Specifies foreground color of the icon. **Note:** You can use the `-foreground` parameter only if you are adding an element of type `toolbutton`. *Data_type*: string, optional
`-icon` *`filename`*	When adding a toolbutton, specifies the file name of the toolbutton icon. *`filename`* can be the full path of the icon file or an icon name supported by Innovus. The icon names used in Innovus can be queried using the `gui_get_ui` command. For list of supported image formats for the icon file, see the "Supported Image Formats" section of the Customizing the User Interface chapter in the *User Guide*. **Note:** The `-icon` parameter is applicable only for elements of type `toolbutton`. *Data_type*: string, optional
`-in` *`name`*	Specifies the name of the parent element. Use this parameter along with `-before` to specify the location of the element you are adding. By default, `-in` adds the element to the end of the parent element. Use `-before` to add the element at a specific location within the parent element. **Note:** If `-in` is not specified, the new element is created but is not added to any location. *Data_type*: string, optional
`-label` *`string`*	Specifies the label to be displayed on the new element. **Note:** This parameter is NOT applicable for elements of type `separator`. *Data_type*: string, optional
`-movable {true | false}`	When adding a toolbar, specifies whether or not it will be movable. *Data_type*: bool, optional

`-newline {true | false}`	When adding a toolbar, specifies whether or not the toolbar should be added as a new row. **Note:** The `-newline` parameter is applicable only for elements of type `toolbar`. *Default*: `false` *Data_type*: bool, optional
`-shortcut` *string*	Specifies the shortcut key for a command, check box, radio button, or toolbutton. **Note:** You can use the `-shortcut` parameter only if you are adding elements of type `command`, `check`, `radio`, or `toolbutton`. *Data_type*: string, optional
`-tooltip` *string*	Specifies the tooltip text for the toolbar or toolbutton you are adding. **Note:** The `-tooltip` parameter is applicable only for elements of type `toolbar` and `toolbutton`. *Data_type*: string, optional
`-type {menu | toolbar | submenu | command | check | radio | separator | toolbutton}`	
	Specifies the type of the interface element you want to add. Currently, elements of type `main` and `statusbar` are not supported. *Data_type*: enum, required
`-underline` *integer*	Specifies the integer index of the character to be underlined in the element label. The index begins with `0` and should have a value less than or equal to the length of the element label. For instance, if you want the second letter of the element label to appear underlined, specify *integer* as `1`. **Note:** The `-underline` parameter is NOT applicable for elements of type `toolbar, separator,` and `toolbutton`. **Note:** This parameter can be used only if the `-label` parameter is also specified. *Data_type*: int, optional

`-value` *`stringValue`*	When adding a radio button, specifies the value to be stored in the associated variable when the radio button is selected. **Note:** The `-value` parameter is applicable only for elements of type `radio`. *Data_type*: string, optional
`-variable` *`string`*	When adding a check box or radio button, specifies the Tcl global variable for the item. *Data_type*: string, optional
`-visible_variable` *`string`*	Represents the Tcl global variable for Visible status *Data_type*: string, optional

Examples

- The following command will add a new menu labeled *ExampleMenu* in the main window:
  ```
  gui_add_ui exmpMenu -type menu -label ExampleMenu -in main
  ```
- The following command will add a new menu item labeled *ExampleCommand* in *ExampleMenu*:
  ```
  gui_add_ui exmpCommand -type command -label ExampleCommand -in exmpMenu -command [list puts Example Command]
  ```
- The following command adds a new, unmovable toolbar to the layout window:
  ```
  gui_add_ui newTB -type toolbar -in main.layout -movable false
  ```

Related Information

- `gui_delete_ui`
- `gui_find_ui`
- `gui_get_ui`
- `gui_set_ui`
- Customizing the User Interface chapter in the *User Guide*
  - *Creating a New Menu*

- *Adding a Menu Element to an Existing Menu*
- *Replacing an Existing Menu Element*
- *Adding a New Toolbar or Toolbutton*

gui_attach_to_cursor

```
gui_attach_to_cursor
[-help]
[ -selected  | objects ]
```

Attaches selected or specified objects to the cursor. The use model is as follows:

1. Select the object(s) to be moved by using the `gui_attach_to_cursor` command.
2. Zoom into the location where you want to place the object(s).
3. Click at the required location to place the selected object(s) there.

This command is useful during floorplanning when you may need to move multiple objects into the same zoomed in area.

Parameters

`-help`	Prints a brief description of the `gui_attach_to_cursor` command. For a detailed description of the command, use the `man` command: `man gui_attach_to_cursor`.
`objects`	Specifies the object pointer list. *Data_type*: (any_object)+, required
`-selected`	Attaches the selected objects to the cursor. *Data_type*: bool, required

Example

The following command attaches objects `A` and `B` to the cursor:

```
gui_attach_to_cursor {A B}
```

Now, move the cursor to the layout and click at the desired location to place objects `A` and `B` there.

Related Information

- gui_select
- gui_zoom

gui_browse_db

```
gui_browse_db
[-help]
start
```

Launches the database browser (DB Browser). The DB browser returns information about one of the following:

- Specified object list
- Top cell of the design
- Selected object

Parameters

`-help`	Prints a brief description that includes type and default information for each `browse_db` parameter. For a detailed description of the command and all of its parameters, use the man command: `man browse_db`
start	Specifies the objects whose attributes need to be displayed in the browser. If you specify an object list, the DB browser displays the attributes of the specified objects in the browser. Use the *Show next object in list* and the *Show last object in list* buttons in the browser to traverse through the list. Alternatively, you can specify one of the following values: • `top`: Displays attributes of the top cell of the design in the browser • `selected`: Displays attributes of the objects you have selected in the main window. Use the *Show next object in list* button in the DB Browser to move to the next object in the sequence. *Data_type*: string, required

Examples

- The following command launches the DB Browser and returns attributes of the top cell of the design.

```
gui_browse_db top
```

Related Information

- Edit Menu chapter in the *Menu Reference*
    - DB Browser

gui_clear_highlight

```
gui_clear_highlight
[-help]
[ -all | obj_list |  -select | -original | -index indexValue]
```

Removes highlights from the highlighted objects at any stage of the design flow. You can choose to clear highlight from all objects, specified objects, selected objects, objects highlighted with original color, or objects with specified index values. If no parameter is specified, all highlights are removed.

Parameters

`-help`	Prints a brief description that includes type and default information for each `gui_clear_highlight` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man gui_clear_highlight`
`-all`	Specifies that highlight will be cleared from all objects. This is the default option. *Data_type*: bool, optional
`-index` *`indexValue`*	Specifies that highlight will be cleared from the objects highlighted with the specified index value. The index value is the value of the Highlight Set as specified through the *View - Highlight Selected* menu or through the `gui_highlight` command. For example, if you specify a value of 3 for index, highlights will be cleared from from all objects that are highlighted as Highlighted Set 3. *Data_type*: int, optional, min=1, max=1024
`obj_list`	Specifies the object pointer list for clearing highlight. *Data_type*: (any_object)+, optional
`-original`	Specifies that highlight should be cleared from the objects highlighted with original color. *Data_type*: bool, optional
`-select`	Specifies that highlight should be cleared from all selected objects. *Data_type*: bool, optional

Examples

- The following command clears highlight from all highlighted objects.
  `gui_clear_highlight`
- The following command clears highlight from all objects that have been highlighted as Highlight Set 4 through the *View - Highlight Selected* menu or through the `gui_highlight`

command.

```
gui_clear_highlight -index 4
```

- The following command clears highlight from all highlighted objects that are currently selected in the display window.

```
gui_clear_highlight -select
```

Related Information

- `gui_highlight`
- `gui_highlight_pin`
- View Menu chapter in the *Menu Reference*:
  - *Highlight Selected*
  - *Clear Highlight*
  - *Edit Highlight Color*

gui_clear_hinst_colors

```
gui_clear_hinst_colors
[-help]
[-color_id colorID [-hinst hInstName]]
```

Resets the color ID for all or the specified hierarchical instances.

Parameters

`-help`	Prints a brief description that includes type and default information for each `gui_clear_hinst_colors` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man gui_clear_hinst_colors`.
`-color_id` *colorID*	Specifies the color ID to which all hierarchical instances or the specified hierarchical instance will be reset. The valid values of color ID are from 0 to 29. The color value for the color ID is the same as specified in the Module Color Preference form (accessed by clicking *Floorplan - Edit Floorplan - Color Module*). *Default*: If no color ID and no instance name is specified, all hierarchical instances are displayed with the color ID `0`. This is the same behavior as the *Reset* button on the Module Color Preference form. *Data_type*: int, optional, min=0, max=29
`-hinst` *hInstName*	Specifies the name of hierarchical instance that should be displayed in the color corresponding to the specified color ID. *Default*: If no instance is specified, the color ID for all hierarchical instances is reset. *Data_type*: string, optional

Example

- The following command resets the color ID for all hierarchical instances and their descendants to `3`:

  ```
  gui_clear_hinst_colors -color_id 3
  ```

- The following command resets the color ID for the hierarchical instance *DTMF_INST* and all its descendants to the color ID 3:

```
gui_clear_hinst_colors -color_id 3 -hinst DTMF_INST
```

Related Information

- gui_set_select_hinst_color
- write_hinst_colors
- Floorplan Menu chapter in the *Menu Reference*
  - *Color Module*

gui_clear_trace_flightline

`gui_clear_trace_flightline [-help]`

Clears all flightlines traced through the `gui_trace_flightline` command.

Parameters

`-help`	Prints a brief description of the `gui_clear_trace_flightline` command. For a detailed description of the command, use the `man` command: `man gui_clear_trace_flightline`.

Related Information

- `gui_report_trace_flightline`
- `gui_trace_flightline`
- `gui_write_flightline`

gui_close_cell_view

```
gui_close_cell_view [-help]
```

Closes the Cell Viewer.

If the Cell Viewer is open at any stage of the design flow, you can use the `gui_close_cell_view` command to close it.

Parameters

`-help`	Prints a brief description of the `gui_close_cell_view` command. For a detailed description of the command, use the `man` command: `man gui_close_cell_view`.

Related Information

- `gui_open_cell_view`
- Tools Menu chapter of the *Menu Reference*
  - *Cell Viewer*

gui_create_floorplan_snapshot

```
gui_create_floorplan_snapshot
[-help]
-dir string
-name string
[-overwrite]
```

Creates snapshots of the design for all three views (Floorplan, Amoeba, and Placement) at any stage of the design flow. The command creates GIF images, for all the design views, on the same scale as can be seen in Innovus. It can also save QoR parameters like WNS,TNS, Placement Overlap, and Congestion at any stage of the design flow. Once the snapshot is created, you can view it using `gui_show_floorplan_snapshot`.

Parameters

`-help`	Prints a brief description that includes type and default information for each `gui_create_floorplan_snapshot` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man gui_create_floorplan_snapshot`.
`-dir string`	Specifies the directory in which you want to save the snapshot. *Data_type*: string, required
`-name string`	Specifies the name of snapshot and the files will be created, like `ss_string_view.gif` for all three views. *Data_type*: string, required
`-overwrite`	Overwrites the existing snapshot with new one. *Data_type*: bool, optional

Note: This command will not work in the `-nowin` mode.

Examples

- The following command will create a design snapshot in the directory `/work/user1`. Before creating the snapshot, it will create the following files:

    - `ss_cts.fplan.gif` for floorplan view
    - `ss_cts.amoeba.gif` for amoeba view
    - `ss_cts.place.gif` for place view

    ```
    gui_create_floorplan_snapshot -dir /work/user1 -name cts
    ```

- The following command will create a snapshot under the name `cts`. If there exists a snapshot under this name in the directory `/work/user1`, it will overwrite it.

  ```
  gui_create_floorplan_snapshot -dir /work/user1 -name cts -overwrite
  ```

Related Information

- `gui_write_picture`
- `gui_show_floorplan_snapshot`
- `gui_write_flow_gifs`
- `write_to_gif`
- Tools Menu chapter of *Menu Reference*
    - *Create Snapshot*
    - *View Snapshot*

gui_create_user_bundle_net

```
gui_create_user_bundle_net
[-help]
name
-end {inst|hinst|port|base_pin|group}
-nets net_list
-start {inst|hinst|port|base_pin|group}
```

Creates a bundle of specified nets that can be displayed as flightlines. Bundled nets help you with data flow analysis before and after layout. Net bundles can be created and viewed in floorplan and amoeba views. Each custom net bundle displays the number of nets in it and works for macros, blobs, and giudes.

Using `gui_create_user_bundle_net`, you can create custom bundled nets that allow you to define your own net bundles as per your requirements. This is useful while debugging physical properties of interconnects within a complicated hierarchical design because you can define start and end points of a net bundle. As a result, you can create and display only relevant flightlines, instead of looking at all the flightlines of specified nets.

Note: One net can be part of multiple net bundles simultaneously.

You can display flightline net bundles using the `gui_show_user_bundle_nets` command and delete them using the `gui_delete_user_bundle_net` command.

Note: You can choose to display flightlines for only bundled nets by using the *Show Net Bundle FlightLine Only* option in the Preferences form. If this option is not selected, the net bundle flightlines are displayed along with other selection-based flightlines by default.

Parameters

`-help`	Prints a brief description that includes type and default information for each `gui_create_user_bundle_net` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man gui_create_user_bundle_net`
`-end {inst|hinst|port|base_pin|group}`	
	Specifies the name of object which is the end point of the nets. The object type includes inst, hinst, port, base_pin, and group. *Data_type*: (inst|hinst|port|base_pin|group), required
name	Specifies the name of the custom net bundle. *Data_type*: string, required
`-nets` *net_list*	Specifies the list of nets to be included in the net bundle. You can use patterns with wildcards to specify net names. All the nets that match the specified pattern are put in the same bundle. *Data_type*: (net)+, required
`-start {inst|hinst|port|base_pin|group}`	
	Specifies the name of object which is the start point of the nets. The object type includes inst, hinst, port, base_pin, and group. *Data_type*: (inst|hinst|port|base_pin|group), required

Examples

- The following command creates a bundle called `bundle_1`, which contains flightlines of nets `net_1` and `net_2`:

  ```
  gui_create_user_bundle_net bundle_1 -nets {net_1 net_2} -start Module1 -end
  Module2
  ```

  You can then use the gui_show_user_bundle_nets command to display `bundle_1` flightlines as follows:

  ```
  gui_show_user_bundle_nets -net_bundle bundle_1
  ```

Related Information

- `gui_delete_user_bundle_net`
- `gui_show_user_bundle_nets`
- View Menu chapter of the *Menu Reference*
  - *Preferences - Flightline*

gui_delete_objs

```
gui_delete_objs
[-help]
[-shape | -text | -marker | -all | -selected ]
```

Deletes specified or selected GUI objects.

Parameters

`-help`	Prints out the command usage. For a detailed description of the command, use the `man` command: `man gui_delete_objs`
`-all`	Deletes all GUI objects. *Data_type*: bool, optional
`-marker`	Deletes GUI markers. *Data_type*: bool, optional
`-selected`	Deletes selected GUI objects. *Data_type*: bool, optional
`-shape`	Deletes GUI shapes. *Data_type*: bool, optional
`-text`	Deletes GUI texts. *Data_type*: bool, optional

Related Information

- `create_gui_shape`
- `create_gui_text`
- `gui_add_marker`

gui_delete_ui

`gui_delete_ui [-help] name`

Deletes an existing interface element with the specified name. If multiple interface elements are found with the same name, the first matched element is deleted. Use this command to customize the Innovus graphical interface by removing menus, menu items, and other interface elements that you do not require any longer.

Parameters

`-help`	Prints a brief description that includes type and default information for each `gui_delete_ui` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man gui_delete_ui`
`name`	Specifies the internal name of the interface element as a string. An error is returned if an interface element with the specified name does not exist. **Note:** Here, *`name`* represents the internal name assigned to the element when it was added to the interface and not the label that is displayed on the interface. If you specify the label instead of the name, the software will return an error. *Data_type*: string, required

Example

- The following command will delete an interface element with the internal name *`Menu1`*:

  ```
  gui_delete_ui Menu1
  ```

Related Information

- `gui_add_ui`
- `gui_find_ui`
- `gui_get_ui`
- `gui_set_ui`
- Customizing the User Interface chapter in the *User Guide*
    - *Replacing an Existing Menu Element*

gui_delete_user_bundle_net

```
gui_delete_user_bundle_net
[-help]
name
```

Deletes the specified flightline net bundle. If a net is part of multiple net bundles and you delete one of the net bundles, other net bundles containing the net are not affected.

You can create flightline net bundles using the `gui_create_user_bundle_net` command and display them using the `gui_show_user_bundle_nets` command.

Note: `gui_delete_user_bundle_net` removes the specified net bundle. If you want to retain the net bundle but just want to clear its flightlines from the display area, use the `-remove` option of `gui_show_user_bundle_nets` instead.

Parameters

`-help`	Prints a brief description that includes type and default information for each `gui_delete_user_bundle_net` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man gui_delete_user_bundle_net`
name	Specifies the name of the flightline net bundle to be removed. *Data_type*: string, required

Examples

- The following command deletes the flightline net bundle `bundle_1`:

  ```
  gui_delete_user_bundle_net bundle_1
  ```

Related Information

- `gui_create_user_bundle_net`
- `gui_show_user_bundle_nets`
- View Menu chapter of the *Menu Reference*
    - *Preferences - Flightline*

gui_deselect

```
gui_deselect
[-help]
{-all  |
-rect {x1 y1 x2 y2} |
-point {x y} |
-line {x1 y1 x2 y2}}
```

Deselects objects as per the parameters you specify. If `-all` is specified, deselects all selected objects.

Parameters

`-help`	Prints a brief description that includes type and default information for each `gui_deselect` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man gui_deselect`
`-all`	Deselects all objects. *Data_type*: bool, optional
`-line {x1 y1 x2 y2}`	Deselects objects through which the specified line passes. *Data_type*: rect, optional
`-point {x y}`	Deselects objects at the specified point. *Data_type*: point, optional
`-rect {x1 y1 x2 y2}`	Deselects objects within the specified box coordinates. *Data_type*: rect, optional

Related Information

- `deselect_obj`
- `gui_select`
- `select_obj`
- The Main Window chapter in *Menu Reference*
    - *Tool Box Widgets*

gui_dim_foreground

```
gui_dim_foreground
[-help]
-light_level {default medium dark}
```

Dims the display area to the specified level. Use this command to view selected and highlighted objects clearly against the dimmed foreground. This command is especially useful when you are debugging.

The equivalent bindkey is F12.

Parameters

`-help`	Prints a brief description that includes type and default information for each `gui_dim_foreground` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man gui_dim_foreground`.
`-light_level {default medium dark}`	
	Specifies the brightness level of the foreground in the display area. *Data_type*: enum, required

Related Information

- `gui_highlight`
- `gui_select`
- View Menu chapter in *Menu Reference*
  - *Dim Background*

gui_find_ui

```
gui_find_ui
[-help]
[name]
[-before name]
[-checked true | false]
[-command string]
[-disabled true | false]
[-enable_variable string]
[-foreground string]
[-icon filename]
[-in parent_name]
[-label string]
[-movable true | false]
[-newline true | false]
[-shortcut string]
[-tooltip string]
[-type {menu toolbar submenu command check radio separator toolbutton}]
[-underline index]
[-value string]
[-variable string]
[-visible_variable string]
```

Enables you to retrieve an element's name by specifying a property-value combination.

Parameters

`-help`	Prints a brief description that includes type and default information for each `gui_find_ui` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man gui_find_ui`
`name`	Specifies the name of the parent element to define the search scope. If multiple elements are found with the same *`name`*, the first matched element is searched. If *`name`* is not specified, a global search is performed. *Data_type*: string, optional

`-before` *name*	Returns the element that is positioned just before the element identified by *name*. *Data_type*: string, optional
`-checked {true | false}`	Returns the names of check boxes and radio buttons with the specified checked state. *Data_type*: bool, optional
`-command` *string*	Returns the name of the interface element that executes the specified command string. **Note:** You can use the `-command` parameter only for elements of type `command, check, radio` or `toolbutton`. *Data_type*: string, optional
`-disabled {true | false}`	Returns the name of the interface elements with the specified state. **Note:** You can use the `-disabled` parameter only for elements of type `menu, submenu, command, check, radio` or `toolbutton`. *Data_type*: bool, optional
`-enable_variable` *string*	Returns the name of the interface elements with the specified Enable status. *Data_type*: string, optional
`-foreground` *string*	Returns the name of the tool button with the specified foreground color of the icon. **Note:** You can use the `-foreground` parameter only for elements of type `toolbutton`. *Data_type*: string, optional

`-icon` *`filename`*	Returns the name of the tool button with the specified image file in its icon. *`filename`* can be the full path of the icon file or an icon name supported by Innovus. The icon names used in Innovus can be queried using the `gui_get_ui` command. **Note:** The `-icon` parameter is applicable only for elements of type `toolbutton`. *Data_type*: string, optional
`-in` *`parent_name`*	Returns the name of the interface elements with the specified parent name. *Data_type*: string, optional
`-label` *`string`*	Returns the name of the interface element with the specified label. **Note:** This parameter is NOT applicable for elements of type `separator`. *Data_type*: string, optional
`-movable true | false`	Returns whether the object is movable or not. This parameter is applicable only for elements of type `toolbar`. *Data_type*: bool, optional
`-newline {true | false}`	Returns the name of the toolbars with the specified newline status. **Note:** The `-newline` parameter is applicable only for elements of type `toolbar`. *Data_type*: bool, optional
`-shortcut` *`string`*	Returns the name of the interface elements with the specified shortcut key. **Note:** You can use the `-shortcut` parameter only for elements of type `command`, `check`, `radio`, or `toolbutton`. *Data_type*: string, optional

`-tooltip` *`string`*	Returns the name of the toolbar or toolbutton with the specified tooltip text. **Note:** The `-tooltip` parameter is applicable only for elements of type `toolbar` and `toolbutton`. *Data_type*: string, optional
`-type {menu | toolbar | submenu | command | check | radio | separator | toolbutton}`	
	Returns the name of the interface elements of the specified type. *Data_type*: enum, optional
`-underline` *`index`*	Returns the name of the interface elements that has character with the specified integer index underlined in the element label. The index begins with `0`, so for example, to find elements that have the third character in their label underlined, specify an index value of 2. *Data_type*: int, optional
`-value` *`stringValue`*	Returns the name of the radio button with the specified value. **Note:** The `-value` parameter is applicable only for elements of type `radio`. *Data_type*: string, optional
`-variable` *`string`*	Returns the name of the check boxes or radio buttons with the specified the Tcl global variable. *Data_type*: string, optional
`-visible_variable` *`string`*	Returns the name of the interface elements with the specified Visible status. *Data_type*: string, optional

Examples

- The following command returns the names of all the menus in the main window:

  ```
  gui_find_ui main -type menu
  ```

  The software returns the following:

  ```
  file view edit partition fplan power place eco clock route timing verify pvs tools
  windows flow help
  ```

- The following command returns the name of the element before the Help menu in the main window:

  ```
  gui_find_ui -before help
  ```

  The software returns the following:

  ```
  flow
  ```

- The following command returns the name of the element with a label "*CCOpt Clock Tree Debugger...*" on the menu in the main window:

  ```
  gui_find_ui -label "CCOpt Clock Tree Debugger..."
  ```

  The software returns the following:

  ```
  _CCOpt_Clock_Tree_Debugger...
  ```

  You can then get any property of the element by using `gui_get_ui`. For example:

  ```
  gui_get_ui _CCOpt_Clock_Tree_Debugger... -command
  ```

  The software returns the following:

  ```
  ctdMainWin::createSetting
  ```

  Here, `ctdMainWin::createSetting` is the command that is invoked when the menu item labeled *CCOpt Clock Tree Debugger...* is clicked.

- The following command returns the name of the element that invokes the command `ctdMainWin::createSetting`:

  ```
  gui_find_ui main -command ctdMainWin::createSetting
  ```

  The software returns the following:

  ```
  _CCOpt_Clock_Tree_Debugger...
  ```

- The following command returns the name of the element with shortcut key F:

  ```
  gui_find_ui -shortcut F
  ```

  The software returns the following:

  ```
  tbView.ac.view.fit
  ```

- The following set of commands can be used to get the names of the items in the Edit -> Bus

Guide submenu:

1. Find the name of the submenu with the label "Bus Guide:
   ```
   gui_find_ui -label "Bus Guide"
   ```
   The software returns the following:
   ```
   edit.busguide
   ```
2. You can then use the following command to find the elements in the "Bus Guide" submenu:
   ```
   gui_find_ui -in edit.busguide
   ```
   The software returns the following:
   ```
   _Edit... _Color _Clear_Color
   ```

- The following command returns all movable toolbars in the main window:
  ```
  gui_find_ui -movable true
  ```
  The software returns the following:
  ```
  tbFile tbEdit tbView tbTools tbReport tbToolBox we main.layout.selectbar
  ```

Related Information

- `gui_add_ui`
- `gui_delete_ui`
- `gui_get_ui`
- `gui_set_ui`
- Customizing the User Interface chapter in the *User Guide*
  - *Querying and Configuring Interface Elements*

gui_fit

```
gui_fit [-help]
```

Fits the entire design in the viewable window. You can use this command at any stage of the design flow to view the design in entirety.

Parameters

`-help`	Prints a brief description of the `gui_fit` command. For a detailed description of the command, use the `man` command: `man gui_fit`.

Related Information

- `gui_zoom`
- View Menu chapter of the *Menu Reference*
  - *Fit*

gui_get_box

`gui_get_box`

Returns the coordinates of the box you draw on the user interface.

When you enter this command, the software displays a hint on the status bar prompting you to click on the Main Window to indicate the first point of the box. You are then prompted to click at the second point, which represents the diagonally opposite corner of the box. After you click the mouse twice, the software returns the coordinates of the box defined by the two corners you specified. The coordinates are returned in mircon units in the format *`x1 y1 x2 y2`*.

Parameters

`-help`	Prints a brief description of the `gui_get_box` command. For a detailed description of the command, use the `man` command: `man gui_get_box`.

Related Information

- `gui_get_coord`
- `gui_get_coords`
- `gui_view_box`

gui_get_coord

```
gui_get_coord [-help]
```

Returns the coordinates of the point you click on the user interface.

When you enter this command, the software displays a hint on the status bar prompting you to click at a point on the Main Window. After you click the mouse, the software returns the `x` and `y` coordinates of the point you specified. The coordinates are returned in mircon units, such as 675.423 461.9775.

Parameters

`-help`	Prints a brief description of the `gui_get_coord` command. For a detailed description of the command, use the `man` command: `man gui_get_coord`.

Related Information

- `gui_get_box`
- `gui_get_coords`

gui_get_coords

```
gui_get_coords [-help]
```

Returns the coordinates of the points you click on the user interface.

When you enter this command, the software displays a hint on the status bar prompting you to click at the first point on the Main Window. You are then prompted to click at subsequent points or press `Esc` to stop. After you press the `Esc` key, the software returns the coordinates of all the points you clicked in micron units in the format *`x1 y1 x2 y2 x3 y3...`*

Parameters

`-help`	Prints a brief description of the `gui_get_coords` command. For a detailed description of the command, use the `man` command: `man gui_get_coords`.

Example

The following command returns coordinates of the three points you click in the UI:

```
> gui_get_coords
```

On the status bar, the following message is displayed:

```
Enter Lines: Point 1. Press ESC to stop
```

On clicking the first point, the status bar message changes to:

```
Enter Lines: Point 2. Press ESC to stop
```

On clicking the second point, the status bar message changes to:

```
Enter Lines: Point 3. Press ESC to stop
```

After clicking all the required points, press ESC to stop. The coordinates of the three points you clicked are returned at the command prompt:

```
0.007 0.05 0.038 0.05 0.038 0.02
```

Related Information

- gui_get_box
- gui_get_coord

gui_get_ui

```
gui_get_ui
[-help]
[name]
[-quiet]
[ -before
 | -checked
 | -children
 | -command
 | -disabled
 | -dock_widget
 | -enable_variable
 | -foreground
 | -geometry
 | -icon
 | -in
 | -label
 | -menu
 | -message
 | -movable
 | -newline
 | -shortcut
 | -statusbar
 | -title
 | -toolbar
 | -tooltip
 | -type
 | -underline
 | -value
 | -variable
 | -visible
 | -visible_variable ]
```

Retrieves the current value of the specified property of the element that is being queried. Using the `gui_get_ui` command, you can query all the properties in the `gui_add_ui` command for each type. In addition, depending on the `type`, some special properties are supported too as described below:

- For elements of type `menu/submenu/toolbar`:
    - `gui_get_ui` *name* `-children`
      Returns a list of all the child elements of the specified interface element.

- For `main` (main window):
    - `gui_get_ui main -menu`
      Returns a list of all the menus in the main window.
    - `gui_get_ui main -toolbar`
      Returns the list of all the toolbars in the main window.
    - `gui_get_ui main -statusbar`
      Returns the status bar in the main window.
    - `gui_get_ui main -geometry`
      Returns the geometry of the main window in the format *`width`*x*`height`*+*`x`*+*`y`*.
    - `gui_get_ui main -title`
      Returns the title of the main window.
    - `gui_get_ui main -closecmd`
      Returns the close confirmation command of the main window.
    - `gui_get_ui main -visible`
      Returns the visibility status of the main window.
- For `statusbar`:
    - `gui_get_ui name -message`
      Returns the current message of the status bar.
    - `gui_get_ui name -visible`
      Returns the visibility status of the status bar.
    - `gui_get_ui name -in`
      Returns the parent element of the status bar.

`statusbar` also supports the `-enable_variable` and `-visible_variable` parameters.

Parameters

`-help`	Prints a brief description that includes type and default information for each `gui_get_ui` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man gui_get_ui`
`name`	Specifies as a string the name of the interface element you want to query. If multiple widgets are found with the same *`name`*, the first matched widget is queried. Returns an error if there is no element with the specified name. *Data_type*: string, optional
`-before`	When specified, returns the element that occurs before the element identified by *`name`*. *Data_type*: string, optional
`-checked`	Returns the on/off state of the specified check box or radio button. If the check box or radio button is selected, returns `true`. If it is deselected, returns `false`. *Data_type*: bool, optional
`-children`	Returns child elements of specified interface element. **Note:** You can use this parameter with elements of type `menu`, `submenu`, and `toolbar`. *Data_type*: string, optional
`-command`	Returns the Tcl command to be executed when the specified element you are adding is invoked. **Note:** You can use the `-command` parameter only with elements of type `command, check, radio` or `toolbutton`. *Data_type*: string, optional

`-disabled`	When specified, returns whether the specified interface element is enabled (`false`) or disabled (`true`). **Note:** You can use the `-disabled` parameter only with elements of type `menu, submenu, command, check, radio` or `toolbutton`. The parameter is NOT applicable for elements of type `toolbar` and `separator`. *Data_type*: bool, optional
`-dock_widget`	Returns the dock widget in the window. *Data_type*: string, optional
`-enable_variable`	Represents the Tcl global variable for Enable status. *Data_type*: string, optional
`-foreground`	Returns the foreground color of the icon for the specified toolbutton. **Note:** You can use the `-foreground` parameter only with elements of type `toolbutton`. *Data_type*: string, optional
`-geometry`	Returns the width, height, and coordinates of the main window. **Note:** The `-geometry` parameter is applicable only for the main window, which is type `main`. *Data_type*: string, optional
`-icon`	Returns the file name of the specified toolbutton. **Note:** The `-icon` parameter is applicable only for elements of type `toolbutton`. *Data_type*: string, optional
`-in`	Returns the name of the parent element. *Data_type*: string, optional
`-label`	Returns the label of the specified element. **Note:** This parameter is NOT applicable for elements of type `separator`. *Data_type*: string, optional

`-menu`	Returns the names of the menus in the main window. **Note:** This parameter is applicable only for the main window, which is type `main`. *Data_type*: string, optional
`-message`	Returns the message on the status bar. **Note:** This parameter is applicable only for the status bar. *Data_type*: string, optional
`-movable`	Returns whether object is movable or not. *Data_type*: bool, optional
`-newline`	Returns whether or not the specified toolbar starts in a new row. **Note:** The `-newline` parameter is applicable only for elements of type `toolbar`. *Data_type*: bool, optional
`-quiet`	Displays the current settings for the specified parameters in the Tcl list format. *Data_type*: bool, optional
`-shortcut`	Returns the shortcut key for a command, check box, radio button, or toolbutton. **Note:** You can use the `-shortuct` parameter only with elements of type `command`, `check`, `radio`, or `toolbutton`. *Data_type*: string, optional
`-statusbar`	Returns the name of the main window's status bar. **Note:** This parameter is applicable only for the main window (`main`). *Data_type*: string, optional
`-title`	Returns the text on the main window's title bar. **Note:** This parameter is applicable only for the main window (`main`). *Data_type*: string, optional

`-toolbar`	Returns the toolbars in the window. *Data_type*: string, optional
`-tooltip`	Returns the tooltip text for the specified toolbar or toolbutton. **Note:** The `-tooltip` parameter is applicable only for elements of type `toolbar` and `toolbutton`. *Data_type*: string, optional
`-type`	Returns the type of the specified element. *Data_type*: enum, optional
`-underline`	Returns the integer index of the underlined character in the element label. **Note:** The `-underline` parameter is NOT applicable for elements of type `toolbar, separator,` and `toolbutton`. *Data_type*: int, optional
`-value`	Returns the value stored in the associated variable of the specified radio button. **Note:** The `-value` parameter is applicable only for elements of type `radio`. *Data_type*: string, optional
`-variable`	Returns the Tcl global variable for the specified check box or radio button. *Data_type*: string, optional
`-visible`	Returns the visibility status of the element. *Data_type*: string, optional
`-visible_variable`	Represents the Tcl global variable for Visible status. *Data_type*: string, optional

Examples

- The following command returns the names of all the menus in the main window:

  ```
  gui_get_ui main -menu
  ```

  The software returns the following:

  ```
  file view edit partition fplan power place eco clock route timing verify pvs tools
  windows flow help
  ```

- The following command returns the label on element named `verify`:

  ```
  gui_get_ui verify -label
  ```

  The software returns the following:

  ```
  Check
  ```

- The following command returns whether the Select bar in the main layout is movable:

  ```
  gui_get_ui main.layout.selectbar -movable
  ```

Related Information

- `gui_add_ui`
- `gui_delete_ui`
- `gui_find_ui`
- `gui_set_ui`
- Customizing the User Interface chapter in the *User Guide*
  - *Querying and Configuring Interface Elements*

gui_get_visible_bumps

```
gui_get_visible_bumps
[-help]
```

Returns the names of all visible bump cells in the design if some bumps have been hidden in the design.

Parameters

`-help`	Prints a brief description that includes type and default information for each `gui_get_visible_bumps` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man gui_get_visible_bumps`

Examples

- The following command returns the name of all visible bumps in the designs:
  ```
  innovus 17> gui_get_visible_bumps
  ```

Related Information

- `gui_set_visible_bumps`

gui_get_visible_net_groups

```
gui_get_visible_net_groups
[-help]
```

Returns the names of the visible net groups if some net groups have been hidden in the design. If all net groups are visible, this command has no effect.

Parameters

`-help`	Prints a brief description that includes type and default information for each `gui_get_visible_net_groups` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man gui_get_visible_net_groups`

Examples

- The following command returns the name of all visible net groups in the designs:

  ```
  innovus 17> gui_get_visible_net_groups
  test_ng
  ```

Related Information

- `gui_set_visible_net_groups`

gui_get_visible_nets

```
gui_get_visible_nets
[-help]
```

Returns the names of the visible nets if some nets have been hidden in the design. If all nets are visible, this command has no effect.

Parameters

`-help`	Prints a brief description that includes type and default information for each `gui_get_visible_nets` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man gui_get_visible_nets`

Examples

- The following command returns the name of all visible nets in the designs:

  ```
  > gui_get_visible_nets
  ram1/subram02_lower/TESTACT_INST_PERIWDSRAM001PAA512W27C3/tpo_a_9_
  ```

Related Information

- `gui_set_visible_nets`

gui_group_hinst

```
gui_group_hinst
[-help]
[-type {flex_model} | hinst]
```

Moves one level up the hierarchy for the specified module guide and displays the next module. The `gui_group_hinst` command is equivalent to the *Group* widget on the toolbar.

Parameters

`-help`	Prints a brief description that includes type and default information for each `gui_group_hinst` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man gui_group_hinst`
hinst	Specifies the name of the hierarchical instance to be grouped. *Data_type*: string, optional
`-type {`*flex_model*`}`	Specifies the type of objects to be grouped. At present, only the `flex_model` type is supported. *Data_type*: enum, optional

Examples

- The following command goes one level up the specified module:
  `gui_group_hinst DTMF_INST/TDSP_CORE_INST`
- The following command groups all flexModels, including flex_module and flex_instgroup, up to the partition or instance group level:
  `gui_group_hinst -type flex_model`

Related Information

- `gui_ungroup_hinst`
- The Main Window chapter in *Menu Reference*
    - *Toolbar Widgets*

gui_hide

```
gui_hide
[-help]
```

Hides the Innovus main window.

You can run the `gui_show` command to re-display the Innovus main window.

Parameters

`-help`	Prints out the command usage. For a detailed description of the command, use the `man` command: `man gui_hide`

Related Information

- `gui_show`
- The Main Window chapter of the *Menu Reference*
  - *Closing or Hiding the GUI*

gui_highlight

```
gui_highlight
[-help]
[object+]
[-color colorValue | -auto_color  | -original]
[-index indexValue]
[-pattern patternValue | -original]
```

Highlights selected or specified objects with the Highlight Set available through the *View - Highlight Selected* menu. The value for the index corresponds to the Highlight Set that will be applied. For example, a value of 4 for the index specifies that all selected objects will be highlighted with the same color and pattern as available for the *Highlight Set #4* in the *View - Highlight Selected* menu. You can use this command at any stage in the design flow.

You can also provide a specific color and/or pattern to be used. All selected objects will be highlighted with the color and/or pattern that you specify.

If you specify a color and or pattern, the Highlight Set corresponding to the index value will be updated to use the color and pattern specified. For example, if you specify an index value of 5, color `green`, and pattern `backslash`, the *Highlight Set #5* will get updated to have the color `green` and the pattern `backslash`.

You can remove highlight from the highlighted objects through the `gui_clear_highlight` command or through the *View - Clear Highlight* menu.

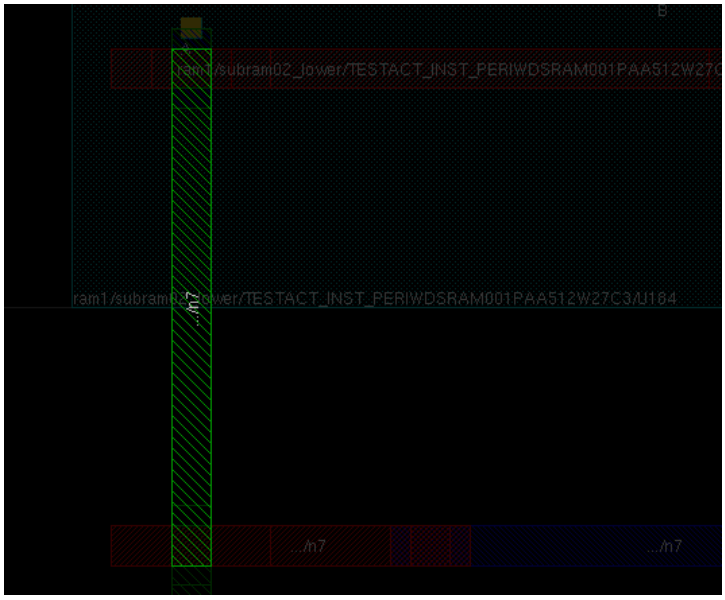
Note: If an object is selectable, you can highlight it using the `gui_highlight` command.

Use the `gui_highlight_pin` command to highlight shapes for specified SIGNAL/POWER pins.

Parameters

`-help`	Prints a brief description that includes type and default information for each `gui_highlight` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man gui_highlight`
object+	Specifies object pointer list for highlighting. *Data_type*: (any_object)+, optional

`-auto_color`	Specifies the object color automatically. When this parameter is specified, the selected object is highlighted with a different color each time you call the `gui_highlight` command. **Note**: This parameter cannot be used with `-index` or `-color`. *Data_type*: bool, optional
`-color` *`colorValue`*	Specifies the color value for the specified object. The color value is the same as that available through the Edit Highlight Color form (*View - Edit Highlight Color*). You can also use the Red Green Blue (RGB) value in the following format: #*`V1V2V3`* where V1, V2, and V3 are the hexadecimal values for the red, green, and blue color components respectively. For example, you can specify the value `#CC66FF`. *Default:* The color defined for the Highlight Set corresponding to the index value is used. *Data_type*: string, optional
`-index` *`indexValue`*	Specifies the index value for highlighting the specified objects. All objects that are selected will be highlighted with the same color and pattern as defined in corresponding Highlight Set in the *View - Highlight Selected* menu. As an example, if you specify a value of 3 for index, all objects that are selected will be highlighted with the same color and pattern as available for the *Highlight Set #3* in the *View - Highlight Selected* menu. Note that the maximum value is related to the *Number of Highlight Color* setting in the Edit Highlight Color form. If this value is 64, the maximum acceptable value for the `-index` option is 64. If set to 256, the maximum acceptable value for `-index` is 256. *Data_type*: int, optional, min=1, max=256

`-original`	Highlights the object in original color. This option is useful if you want to check a specific object more closely with the background dimmed. An object highlighted in original color will stand out in the dimmed background:  **Note**: This parameter cannot be used with `-index`, `-color`, or `-pattern`. *Data_type*: bool, optional
`-pattern` *patternValue*	
	Specifies the pattern value for the specified object. The pattern value is the same as the Stipple Value available through the Edit Highlight Color form. *Default:* The pattern (Stipple Value) for the Highlight Set corresponding to the index value is used. *Data_type*: string, optional

Examples

- The following command highlights all selected objects with the same color and pattern as defined for the *Highlight Set #4* in the *View - Highlighted Selected* menu.

  ```
  gui_highlight -index 4
  ```

- The following command highlights all selected objects with the color `green` and the pattern `dot4`. The index values is set to 5 and, therefore, the command will also change the definition of the *Highlight Set #5* (*View - Highlight Selected* menu) to have color `green` and pattern `dot4`.

  ```
  gui_highlight -index 5 -color green -pattern dot4
  ```

- The following command highlights all selected objects with the color corresponding to the RGB hexadecimal value `#33FF66` and the pattern `horizontal`. The index values is set to 3 and, therefore, the command will also change the definition of the *Highlight Set #5* (*View - Highlight Selected* menu) to have color `#33FF66` and pattern `horizontal`.

  ```
  gui_highlight -index 3 -color #33FF66 -pattern horizontal
  ```

Related Information

- `gui_clear_highlight`
- `gui_highlight_pin`
- View Menu chapter in the *Menu Reference*:
  - *Highlight Selected*
  - *Clear Highlight*
  - *Edit Highlight Color*

gui_highlight_pin

```
gui_highlight_pin
[-help]
-inst string
-pin string
[-original | -index indexValue]
[-original | -color colorValue]
[-original | -pattern patternValue]
```

Highlights the shapes of all specified SIGNAL/POWER instance pins with the Highlight Set available through the *View - Highlight Selected* menu. The value for the index corresponds to the Highlight Set that will be applied. For example, a value of 4 for the index specifies that all selected objects will be highlighted with the same color and pattern as available for the *Highlight Set #4* in the *View - Highlight Selected* menu.

You can also provide a specific color and/or pattern to be used. All specified pins will be highlighted with the color and/or pattern that you specify.

Note: If you specify a color and or pattern, the Highlight Set corresponding to the index value will be updated to use the color and pattern specified. For example, if you specify an index value of `5`, color `green`, and pattern `backslash`, the *Highlight Set #5* will get updated to have the color `green` and the pattern `backslash`.

Parameters

`-help`	Prints a brief description that includes the type and default information for each `gui_highlight_pin` parameter. For a detailed description of the command and all of its parameters, use the man command: `man gui_highlight_pin`

`-color` *colorValue*	Specifies the color value for the specified object. The color value is the same as that available through the Edit Highlight Color form, which can be accessed by selecting *View - Edit Highlight Color*. You can also use the Red Green Blue (RGB) value in the following format: `#`*V1V2V3* where *V1*, *V2*, and *V3* are the hexadecimal values for the Red, Green, and Blue color components, respectively. For example, you can specify the value `#CC66FF`. *Default:* The color defined for the Highlight Set corresponding to the specified index value is used. *Data_type*: string, optional
`-index` *indexValue*	Specifies the index value for the specified object. By default, all specified pins are highlighted with the same color and pattern as defined in the *Highlight Set* corresponding to the specified index value in the *View - Highlight Selected* submenu. For example, if you specify a value of 3 for index, all specified pins are highlighted with the same color and pattern as available for *Highlight Set #3* in the *View - Highlight Selected* submenu. *Data_type*: int, required
`-inst` *string*	Specifies the name of instance. *Data_type*: string, required
`-original`	Highlights the instance pin in original color. This option is useful if you want to check a specific instance pin more closely with the background dimmed. An object highlighted in original color will stand out in the dimmed background. **Note**: This parameter cannot be used with `-index`, `-color`, or `-pattern`. *Data_type*: bool, optional
`-pattern` *patternValue*	Specifies the pattern value for the specified object. The pattern value is the same as the *Stipple* value available through the Edit Highlight Color form. *Default:* The pattern (*Stipple* value) defined for the Highlight Set corresponding to the specified index value is used. *Data_type*: string, optional

`-pin string`	Specifies the name of SIGNAL/POWER pin to be highlighted. You can specify more than one pin name. *Data_type*: string, required

Examples

- The following command highlights the SIGNAL pins `E` and `D` of instance `U1` using the color yellow and the solid pattern:
  ```
  gui_highlight_pin -inst U1 -pin {E D} -color yellow -pattern solid -index 1
  ```
- The following command highlights the SIGNAL pin `E` and the POWER pin `gnd` of instance `U1` using the color yellow and the solid pattern:
  ```
  gui_highlight_pin -inst U1 -pin {E gnd} -color yellow -pattern solid -index 1
  ```

Related Information

- `gui_clear_highlight`
- `gui_highlight`
- `gui_highlight_pin_connection`
- View Menu chapter in the *Menu Reference*:
  - *Highlight Selected*
  - *Clear Highlight*
  - *Edit Highlight Color*

gui_highlight_pin_connection

```
gui_highlight_pin_connection
[-help]
-from_pin pin
[-mode {wire_segment | whole_net | flight_line}]
-to_pin pin
[-with_arrow | -select_only]
[-net_color_index index_value | -select_only]
[-inst_color_index index_value | -select_only]
```

Highlights the full timing path or its segment in the text format while generating the sign-off timing report. Use this command to quickly debug the timing path (timing point information) used in the tool.

Parameters

`-help`	Prints a brief description that includes the type and default information for each `gui_highlight_pin_connection` parameter. For a detailed description of the command and all its parameters, use the `man` command: `man gui_highlight_pin_connection`
`-from_pin pin`	Specifies the name of the starting pin to be highlighted. *Data_type*: (pin), required
`-inst_color_index index_value`	
	Specifies the instance color index value (an integer value from 1 to 64) for the specified timing path. *Data_type*: int, optional, min=1, max=64
`-mode {wire_segment | whole_net | flight_line}`	

	Specifies the draw mode between the starting and ending pins. *Data_type*: enum, optional
`-net_color_index` *index_value*	
	Specifies the net color index value (an integer value from 1 to 64) for the specified timing path. *Data_type*: int, optional, min=1, max=64
`-select_only`	Selects only the specified path for highlighting. *Data_type*: bool, optional
`-to_pin` *pin*	Specifies the name of the ending pin to be highlighted. *Data_type*: (pin), required
`-with_arrow`	Specifies whether or not the connection is defined with an arrow. *Data_type*: bool, optional

Example

The following command highlights a line drawn from the specified input pin to output pin while generating the sign-off timing report:

```
gui_highlight_pin_connection -from_pin ffa/Q -U7/A -mode flight_line
```

The timing report generated is shown below:

```
:::::::Timing Report::::::::::

  Startpoint: ffa (rising edge-triggered flip-flop clocked by CLK)

  Endpoint: ffd (rising edge-triggered flip-flop clocked by CLK)

  Path Group: CLK

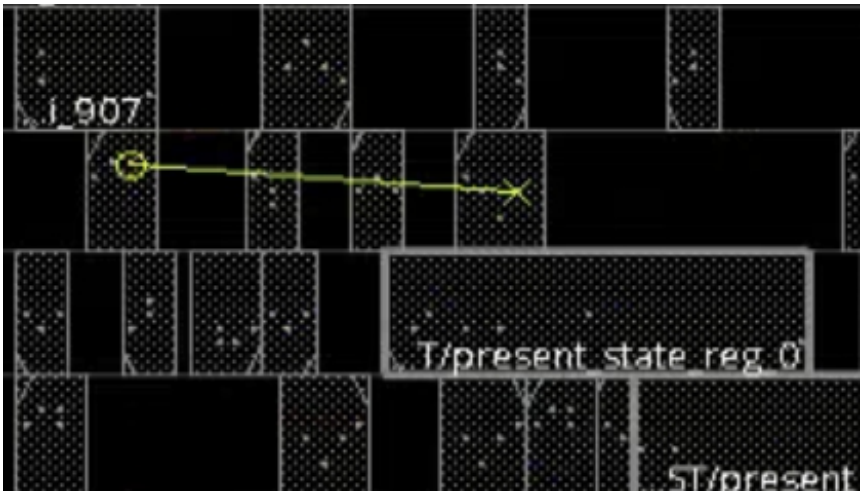
  Path Type: max
```

Point	Fanout	Cap	Trans	Incr	Path
clock CLK (rise edge)				0.00	0.00
clock network delay (propagated)				4.90	4.90
ffa/CLK (DTC10)			0.30	0.10	5.00 r
ffa/Q (DTC10)			0.57	1.70	6.70 f
b (net)	2	3.85			
U7/A (IV110)			0.57	0.00	6.70 f
U7/Y (IV110)			1.32	0.84	7.55 r
QA (net)	50	32.59			
U12/A (NA310)			3.02	4.00	11.55 r
U12/Y (NA310)			2.47	4.04	15.58 f
n9 (net)	3	8.87			
U17/A (NA211)			2.47	0.00	15.58 f
U17/Y (NA211)			1.01	1.35	16.94 f
n14 (net)	2	4.87			
U23/A (IV120)			1.01	0.00	16.94 f
U23/Y (IV120)			0.51	0.37	17.30 r
n13 (net)	1	2.59			
U15/A2 (BF003)			0.51	0.00	17.31 r
U15/Y (BF003)			0.88	0.81	18.12 f
n12 (net)	1	15.61			
U16/B2 (BF003)			3.88	3.00	21.12 f
U16/Y (BF003)			2.46	0.99	22.11 r
n7 (net)	1	2.61			
U10/A (AN220)			2.46	0.00	22.11 r
U10/Y (AN220)			0.46	1.04	23.15 r
n15 (net)	1	2.63			

```
  ffd/D (DTN10)                                  0.46       0.00      23.15 r
  data arrival time                                                   23.15
 clock CLK (rise edge)                                      10.00      10.00
  clock network delay (propagated)                           3.50      13.50
  ffd/CLK (DTN10)                                                      13.50 r
  library setup time                                        -1.33      12.17
  data required time                                                   12.17
  ---------------------------------------------------------------------------
  data required time                                                   12.17
  data arrival time                                                   -23.15
  ---------------------------------------------------------------------------
  slack (VIOLATED)                                                    -10.98

:::::::END:::::::::::
```

The drawn layout along with the highlighted line is shown in the below picture:



gui_list_marker

```
gui_list_marker [-help]
```

Gives a list of GUI markers in the design layout.

Parameters

`-help`	Prints a brief description of the `gui_list_marker` command. For a detailed description of the command, use the `man` command: `man gui_list_marker`.

Example

- The following command shows a list of GUI markers:
  ```
  gui_list_marker
  ```
  The following is displayed:
  ```
  {name:aa  coordinate:(100.000 100.000)}
  ```

Related Information

- `gui_add_marker`
- `gui_remove_marker`

gui_open_cell_view

```
gui_open_cell_view [-help]
```

Opens the Cell Viewer. The Cell Viewer provides a convenient way to view the content of a cell, cell view, or partition without having to zoom into the design. It provides an artwork window for viewing the following objects:

- any LEF library cell
- any via cell
- any OpenAccess database cell view
- abstract view of any partition in the design
- power grid view

You can use the `gui_open_cell_view` command at any stage of the design flow.

Parameters

`-help`	Prints a brief description of the `gui_open_cell_view` command. For a detailed description of the command, use the `man` command: `man gui_open_cell_view`.

Related Information

- `gui_close_cell_view`
- Tools Menu chapter of the *Menu Reference*
  - *Cell Viewer*

gui_open_schematic

```
gui_open_schematic
[-help]
[-name string]
[-type {module cone}]
[-objs <inst | net | pin | port | hinst>+ | -selected]
```

Opens the schematic view of the specified or selected objects.

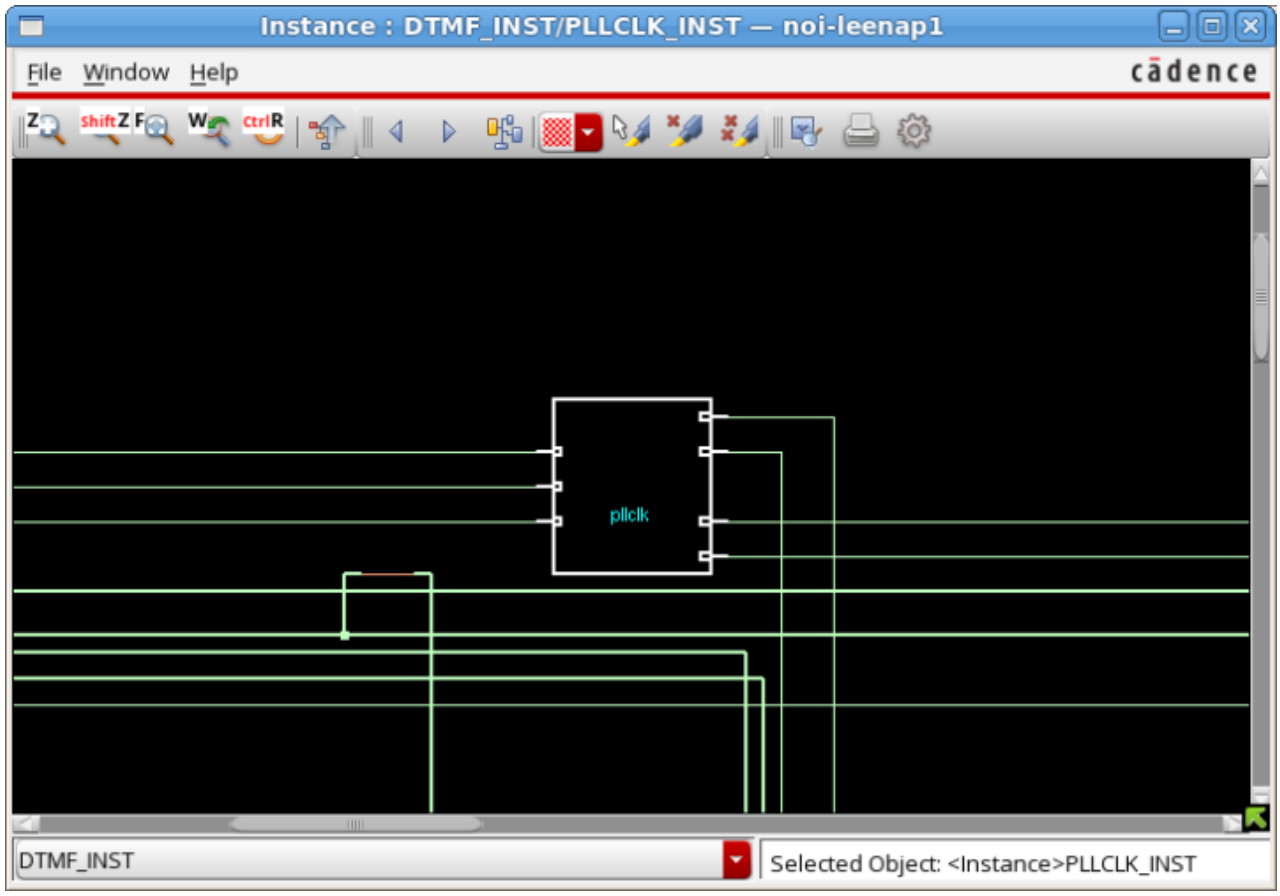
Parameters

`-help`	Prints out the command usage. For a detailed description of the command, use the `man` command: `man gui_open_schematic`
`-name` *`string`*	Specifies a unique window name for the schematic view to be opened. *Data_type*: string, optional
`-objs <inst | net | pin | port | hinst>+`	
	Specifies the object(s) to be added in the schematic view. If multiple objects are specified, the Schematic Viewer can be opened only in the flattened mode (`-type cone`). As the default schematic mode is hierarchical, this means you must always specify `-type cone` when specifying multiple objects. **Note**: This option cannot be used with `-objs` or `-type module`. *Data_type*: (*inst*|*net*|*pin*|*port*|*hinst*)+, optional
`-selected`	Specifies that the selected object(s) are to be added in the schematic view. If multiple objects are selected, the Schematic Viewer can be opened only in the flattened mode (`-type cone`). As the default schematic mode is hierarchical, this means you must always specify `-type cone` when selecting multiple objects. **Note**: This option cannot be used with `-objs` or `-type module`. *Data_type*: bool, optional
`-type {module cone}`	Specifies the type of schematic to be opened. If you specify `cone`, the tool opens the Schematic Viewer to display schematic in the flattened mode. If you specify `module`, the tool opens the Schematic Viewer to display schematic in the hierarchical mode. *Default*: `module` *Data_type*: enum, optional

Examples

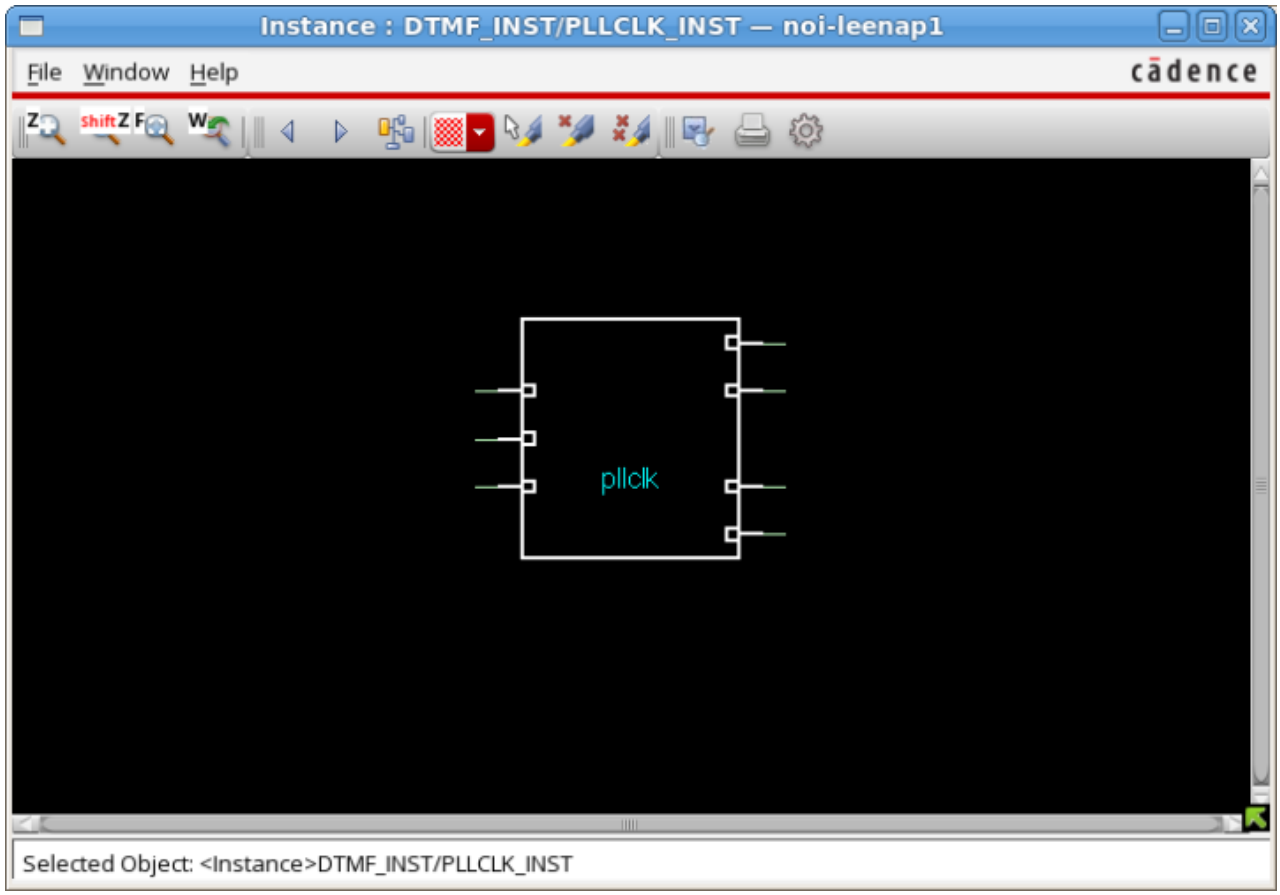
- The following command opens the schematic for the specified instance in the hierarchical mode:

```
gui_open_schematic -objs DTMF_INST/PLLCLK_INST
```



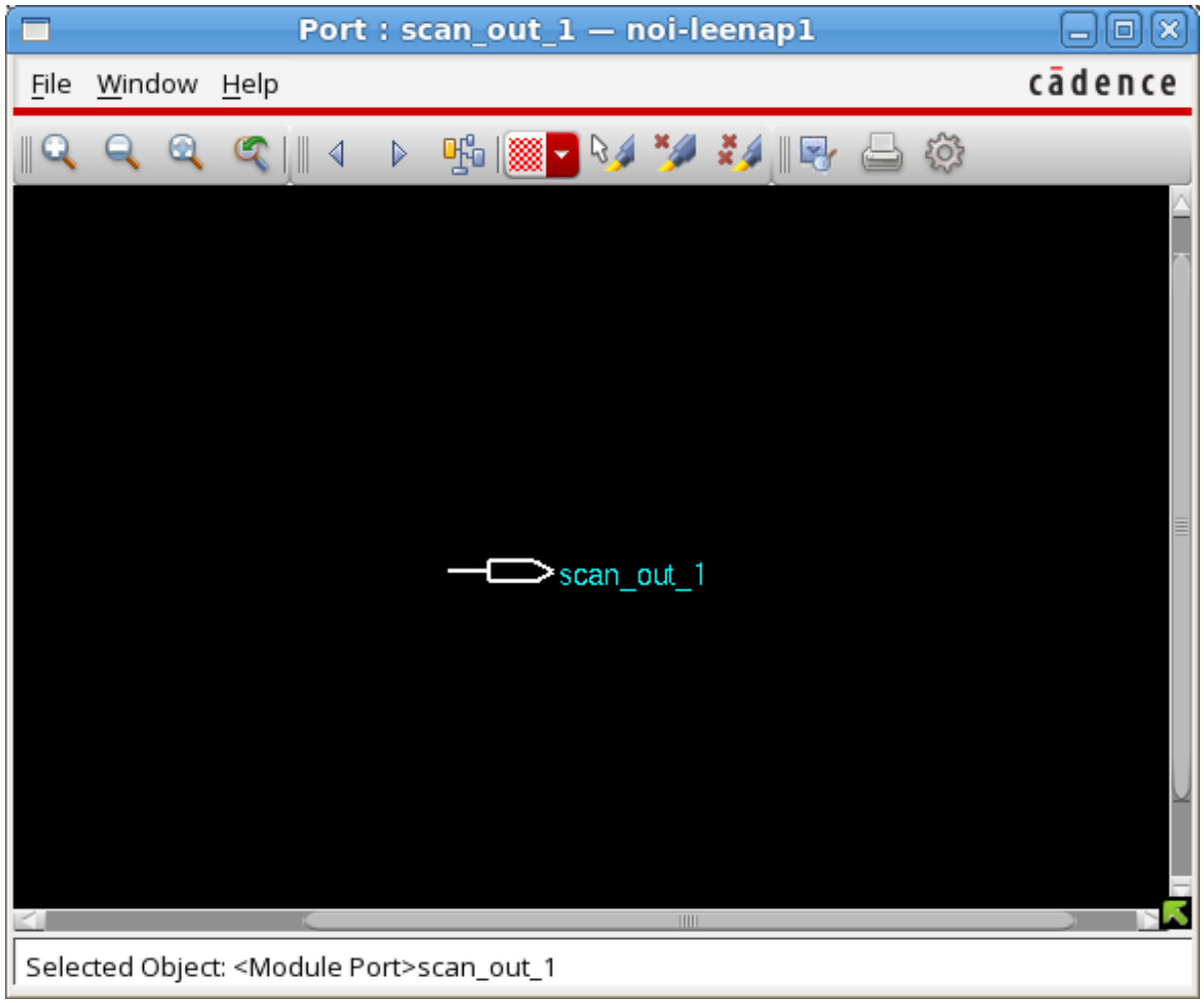
- The following command opens the schematic for the specified instance in the flattened mode:

```
gui_open_schematic -objs DTMF_INST/PLLCLK_INST -type cone
```



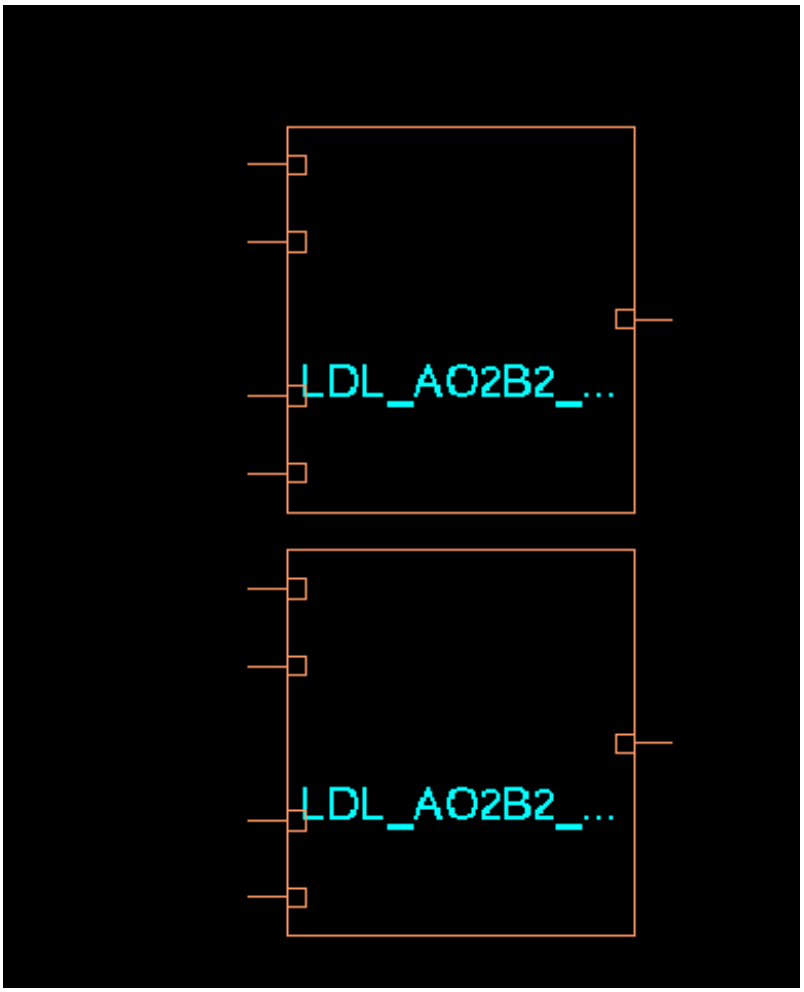
- The following command opens the schematic for the selected object in the flattened mode:

```
gui_open_schematic -selected -type cone
```



- The following command opens the schematic for multiple selected objects in the flattened mode:

```
gui_open_schematic -selected -type cone
```



Related Information

- `gui_schematic_load_view`
- `gui_schematic_save_view`
- Debugging Timing Results chapter in the *User Guide*
  - *Viewing Schematics*

gui_pan

```
gui_pan
[-help]
dx
dy
```

Specifies that the new center of the viewing window should shift from the current center of the viewing window by (*x*, *y*) microns. Use this command at any stage in the design flow to pan or move the design view in a specific direction.

Parameters

`-help`	Prints a brief description that includes type and default information for each `gui_pan` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man gui_pan`.
dx	Specifies that the design center should shift *dx* microns on the x-axis from the center of the viewing window. You can specify positive or negative values for *dx*. A positive value specifies shifting to the right on the x axis. *Data_type*: float, required
dy	Specifies that the design center should shift `dy` microns on the y axis values from the center of the viewing window. You can specify positive or negative values for *dy*. A positive value specifies shifting up on the y axis. *Data_type*: float, required

Example

The following example shifts the design center to the right by 200 microns and down by 300 microns from the center of the viewing window.

```
gui_pan 200 -300
```

Related Information

- `gui_pan_center`
- `gui_pan_page`
- View Menu chapter in the *Menu Reference*
  - *Pan*
- The Main Window chapter in the *Menu Reference*
  - *Mouse Operations*

gui_pan_center

```
gui_pan_center [-help] x y
```

Pans in the viewable window to the center point defined by the specified coordinates. Use this command at any stage of the design flow to view the design area around a specific point.

Parameters

`-help`	Prints a brief description that includes type and default information for each `gui_pan_center` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man gui_pan_center`.
x	Specifies the x coordinate of the center point. *Data_type*: float, required
y	Specifies the y coordinate of the center point. *Data_type*: float, required

Example

The following command pans the viewable window to the center point (x=100, y=200)

```
gui_pan_center 100 200
```

Related Information

- `gui_pan`
- `gui_pan_page`
- View Menu chapter in the *Menu Reference*
  - *Pan*
- The Main Window chapter in the *Menu Reference*
  - *Mouse Operations*

gui_pan_page

`gui_pan_page [-help] x y`

Pans the viewable window in pages defined by the offsets. The size of each page is equal to the size of the current viewable window. Use this command at any stage of the design flow.

Parameters

`-help`	Prints a brief description that includes type and default information for each `gui_pan_page` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man gui_pan_page`.
x	Specifies the `x` offset to indicate number of pages to pan. A positive value of *x* specifies a pan of *x* pages to the right. *Data_type*: float, required
y	Specifies the `y` offset to indicate number of pages to pan. A positive number of *y* specifies a pan of *y* pages toward the top. *Data_type*: float, required

Examples

- The following command pans the display to the right by one page
  ```
  gui_pan_page 1 0
  ```
- The following command pans the display to the left by one page
  ```
  gui_pan_page -1 0
  ```

Related Information

- `gui_pan`
- `gui_pan_center`
- View Menu chapter in the *Menu Reference*
  - *Pan*
- The Main Window chapter in the *Menu Reference*
  - *Mouse Operations*

gui_redo

```
gui_redo [-help]
```

Returns the design to the state it was in immediately prior to issuing the `gui_undo` command. It can only be used to reinstate power planning, bump placement, and selected floorplanning and wire editing functions that were removed with the `gui_undo` command.

You can use this command after you use the `gui_undo` command for any of the following text commands:

- Selected bump placement commands:
  - `create_bump`
  - `delete_bumps`
- Selected floorplanning commands:
  - `align_selected`
  - `flip_or_rotate_obj`
  - `move_obj`
  - `snap_floorplan`
  - `snap_floorplan_io`
  - `space_selected`
- Selected power planning commands:
  - `add_rings`
  - `add_stripes`
  - `update_power_vias`
  - `route_special`
- All wire editor commands:
  - `create_fillet`
  - `convert_special_routes_for_eco`
  - `delete_fillet`
  - `delete_routes`

- deselect_routes
- edit_add_polygon_point
- edit_add_route_point
- edit_add_trim_metal
- edit_create_via
- edit_cut_route
- edit_duplicate_routes
- edit_end_polygon_point
- edit_end_route_point
- edit_merge_routes
- edit_move_routes
- edit_resize_routes
- edit_rotate
- edit_split_routes
- edit_split_wide_routes
- edit_trim_routes
- edit_update_route_layer
- edit_update_route_net
- edit_update_route_rule
- edit_update_route_status
- edit_update_via
- select_routes

Parameters

`-help`	Prints a brief description of the `gui_redo` command. For a detailed description of the command, use the `man` command: `man gui_redo`

Related Information

- `gui_undo`
- Edit Menu chapter in the *Menu Reference*
  - *Undo*
  - *Redo*

gui_redraw

```
gui_redraw [-help]
```

Refreshes the display in the viewable window. This command performs the same function as the *Redraw* icon (CTRL+R keyboard shortcut). You can use this command at any stage in the design flow.

Parameters

`-help`	Prints a brief description of the `gui_redraw` command. For a detailed description of the command, use the `man` command: `man gui_redraw`.

gui_register_edit_callback

```
gui_register_edit_callback
[-help]
-command string
-edit_type {selection pre-move post-move}
```

Adds the callback function to the specified move action. In GUI, the move action involves the following three steps:

1. Click an object.
2. Move the object to the new location.
3. Click to place the object at the new location and complete the move.

The callback function is implemented based on the move action specified with the `-edit_type` parameter.

Parameters

`-help`	Prints the usage of the command. For a detailed description of the command, use the `man` command: `man gui_register_edit_callback`
`-command string`	Specifies the name of the callback function. *Data_type*: string, required
`-edit_type {selection pre-move post-move}`	Specifies the GUI edit function: • `selection` - Implements the callback function when an object is selected (after Step 1). • `pre-move` - Implements the callback function just before the object is placed at the new location (after Step 2, just before you change the coordinates in Step 3). • `post-move` - Implements the callback function after an object is moved (after Step 3, when the tool updates the DB records). *Data_type*: enum, required

Example

Consider the following proc definition:

```
proc move_event_callback {args} {
     set objList $args
     foreach obj $objList {
           set type [dbGet $obj.objType]
           if {$type == "sdp"} {
                custom_snap_sdp $obj
           } elseif {$type == "inst"} {
                set cellName [dbGet $obj.cell.name]
                if {[regexp grf $cellName]} {
                     custom_snap_grf $obj
                }
           }
}
}
```

The following command adds the `move_event_callback` function to the selected object just before the move is completed:

```
gui_register_edit_callback -edit_type pre_move -command {move_event_callback [dbGet selected]}
```

Related Information

- `report_gui_edit_callback`
- `gui_unregister_edit_callback`

gui_remove_marker

```
gui_remove_marker
[-help]
{-name <string> | -all }
```

Removes a marker from the design layout.

Parameters

`-help`	Outputs the command usage and a brief description about the command parameters.
`-all`	Removes all markers from the design layout. *Data_type*: bool, required
`-name <string>`	Specifies the name of the marker to be removed from the design layout. *Data_type*: string, required

Related Information

- `gui_add_marker`
- `gui_list_marker`

gui_report_trace_flightline

```
gui_report_trace_flightline
[-help]
-file report_name
```

Saves information about the all flightlines traced through the `gui_trace_flightline` command to the specified file.

Parameters

`-help`	Prints a brief description of the `gui_report_trace_flightline` command. For a detailed description of the command, use the `man` command: `man gui_report_trace_flightline`.
`-file report_name`	Specifies the name of the report file. *Data_type*: string, required

Example

The following set of commands reports all traced flightlines:

```
innovus 3> gui_trace_flightline -inst  DTMF_INST/DMA_INST/present_state_reg_3 -sink -
sink_level 2 -sink_color pink
innovus 4> gui_trace_flightline -inst DTMF_INST/RESULTS_CONV_INST/r852_reg_5 -sink -
sink_level 2 -sink_color green
innovus 5> gui_report_trace_flightline -file file1
```

```
The tool saves the report as follows:
```

```
#######################################################
# Generated by: Cadence Innovus 18.11-e070_1
# OS: Linux x86_64(Host ID noi-leenap1)
# Generated on: Wed Jul 25 11:51:06 2018
# Design: DTMF_CHIP
# Command: gui_report_trace_flightline
#######################################################


Path0
```

```
Start: DTMF_INST/DMA_INST/present_state_reg_3/Q End:
DTMF_INST/DMA_INST/top_buf_flag_reg/SI
Start: DTMF_INST/DMA_INST/top_buf_flag_reg/Q End: DTMF_INST/TDSP_DS_CS_INST/i_4386/A

Path1
Start: DTMF_INST/DMA_INST/present_state_reg_3/Q End:
DTMF_INST/DMA_INST/top_buf_flag_reg/SI
Start: DTMF_INST/DMA_INST/top_buf_flag_reg/Q End:
DTMF_INST/TDSP_CORE_INST/TDSP_CORE_MACH_INST/samp_bio_reg/D

Path2
Start: DTMF_INST/DMA_INST/present_state_reg_3/Q End:
DTMF_INST/DMA_INST/top_buf_flag_reg/SI
Start: DTMF_INST/DMA_INST/top_buf_flag_reg/Q End: DTMF_INST/DMA_INST/read_spi_reg/SI

....

Path18
Start: DTMF_INST/RESULTS_CONV_INST/r852_reg_5/Q End:
DTMF_INST/RESULTS_CONV_INST/i_1901/B0
Start: DTMF_INST/RESULTS_CONV_INST/i_1901/Y End: DTMF_INST/RESULTS_CONV_INST/i_7337/A1

Path19
Start: DTMF_INST/RESULTS_CONV_INST/r852_reg_5/Q End:
DTMF_INST/RESULTS_CONV_INST/r852_reg_6/SI
Start: DTMF_INST/RESULTS_CONV_INST/r852_reg_6/Q End:
DTMF_INST/RESULTS_CONV_INST/i_1894/B0

Path20
Start: DTMF_INST/RESULTS_CONV_INST/r852_reg_5/Q End:
DTMF_INST/RESULTS_CONV_INST/r852_reg_6/SI
Start: DTMF_INST/RESULTS_CONV_INST/r852_reg_6/Q End:
DTMF_INST/RESULTS_CONV_INST/r852_reg_7/SI

...

Path39
Start: DTMF_INST/RESULTS_CONV_INST/r852_reg_5/Q End:
DTMF_INST/RESULTS_CONV_INST/i_198/AN
Start: DTMF_INST/RESULTS_CONV_INST/i_198/Y End: DTMF_INST/RESULTS_CONV_INST/i_9316/A

Path40
Start: DTMF_INST/RESULTS_CONV_INST/r852_reg_5/Q End:
DTMF_INST/RESULTS_CONV_INST/i_198/AN
Start: DTMF_INST/RESULTS_CONV_INST/i_198/Y End: DTMF_INST/RESULTS_CONV_INST/i_1455/B0
```

Related Information

- gui_clear_trace_flightline
- gui_trace_flightline
- gui_write_flightline

gui_schematic_load_view

```
gui_schematic_load_view
[-help]
-file file_name
[-name name]
```

Loads the specified schematic file.

⚠ This commands work only in the cone view (flattened mode).

Parameters

`-help`	Prints a brief description of the `gui_schematic_load_view` command. For a detailed description of the command, use the `man` command: `man gui_schematic_load_view`.
`-file` *file_name*	Specifies the name of the schematic file to be loaded.
`-name` *name*	Specifies a unique window name for the opened schematic view.

Related Information

- `gui_open_schematic`
- `gui_schematic_save_view`

gui_schematic_save_view

```
gui_schematic_save_view
[-help]
-file file_name
```

Saves the current schematic view into the specified file.

⚠ This commands work only in the cone view (flattened mode).

Parameters

`-help`	Prints a brief description of the `gui_schematic_save_view` command. For a detailed description of the command, use the `man` command: `man gui_schematic_save_view`.
`-file` *`file_name`*	Specifies the name of the file in which to save the schematic view.

Related Information

- `gui_open_schematic`
- `gui_schematic_load_view`

gui_select

```
gui_select
[-help]
{-point {x y} |
-line {x1 y1 x2 y2} |
-rect {x1 y1 x2 y2}}
[-toggle  | -append  | -next]
[-point {x y} [-next ]]
```

Selects objects as per the parameters you specify. Use `-point` to select the object at the specified coordinate, `-line` to select the objects through which the specified line passes, or `-rect` to select the objects falling within the specified box coordinates.

Parameters

`-help`	Prints a brief description that includes type and default information for each `gui_select` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man gui_select`
`-append`	Appends the selection. This is equivalent to pressing the `Shift` key while selecting an object using the GUI. *Data_type*: bool, optional
`-line {x1 y1 x2 y2}`	Selects by line. Use this parameter to select the objects through which the specified line passes. *Data_type*: (point)+, optional
`-next`	Steps to next object under point. This is same as using space bar while selecting an object using the GUI. *Data_type*: bool, optional
`-point {x y}`	Selects the object at the specified point. *Data_type*: point, optional
`-rect {x1 y1 x2 y2}`	Selects by box. Use this parameter to select the objects falling within the specified rectangular area. *Data_type*: rect, optional
`-toggle`	Toggles the selection. This is equivalent to pressing the `Ctrl` key while selecting an object using the GUI. *Data_type*: bool, optional

Examples

- The following command selects the objects that are totally enclosed within the box defined by the coordinates (`x = 10, y = 200`) and (`x = 100, y = 50`):
  ```
  gui_select -rect {10 200 100 50}
  ```
- The following command selects all the objects falling across the specified line coordinates:
  ```
  gui_select -line {168.654 994.049 688.129 436.224}
  ```
- The following command selects the objects crossed by the specified line and appends them to the current selection:
  ```
  gui_select -line {169.622 996.049 698.290 446.240} -append
  ```
- The following command toggles the selection on the objects in the specified bounding box. If the objects are previously selected, the command deselects them and vice versa:
  ```
  gui_select -rect {10 200 100 50} -toggle
  ```

Related Information

- `deselect_obj`
- `gui_deselect`
- `select_obj`
- The Main Window chapter in *Menu Reference*
  - *Tool Box Widgets*

gui_select_highlighted

```
gui_select_highlighted
[-help]
[-type {inst wire via}]
```

Selects highlighted objects. If `-type` is specified, selects highlighted objects of the specified type.

Parameters

`-help`	Prints a brief description that includes type and default information for each `gui_select_highlighted` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man gui_select_highlighted`
`-type {inst wire via}`	Specifies the type of the highlighted objects to be selected. Options are: • `inst` - Selects highlighted instances • `wire` - Selects highlighted wires • `via` - Selects highlighted vias *Data_type*: enum, optional

Examples

- The following command selects all highlighted instances.
  ```
  gui_select_highlighted -type instance
  ```

Related Information

- `gui_highlight`
- `gui_select`

gui_set_obj_color

```
gui_set_obj_color
[-help]
{[-obj_name InstOrHInstName | -obj_type typeList]
[-multicolor | -color_id colorID]
[-include_children {off | color | multicolor}]
[-reset]}
```

Sets the color for objects of the specified name or type. You can choose to include children when specifying object color for a hierarchical instance (HInst).

The `gui_set_obj_color` command is typically used during floorplanning to color objects that are currently displayed in the design display area. You can use this command to distinguish between objects of different type or between objects belonging to different instances, hierarchical instances, instance groups, or power domains.

Before using the `gui_set_obj_color` command, you should have:

- Loaded the floorplan
- Ungrouped objects so that only required objects are displayed in the design display area in the Floorplan or Physical view.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `gui_set_obj_color` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man gui_set_obj_color`.
`-color_id` *`colorID`*	Specifies the color ID from 0 to 29. The color value for the color ID is the same as specified in the Module Color Preference form (accessed by clicking *Floorplan - Edit Floorplan - Color Module*). *Default*: Color ID `0` *Data_type*: int, optional, min=0, max=29

`-include_children {off | color | multicolor}`	Specifies whether the children of the hierarchical instance will be colored simultaneously when you specify a color for an HInst. You can set one of the following options: • `off`: The child instances will not be colored. • `color`: The child instances are colored the same color as their parent instance. • `multicolor`: The child instances are colored in multiple colors. *Data_type*: enum, optional
`-multicolor`	Uses multiple colors to color objects of the specified type, instance, or hierarchical instance. *Data_type*: bool, optional
`-obj_name` *InstOrHInstName*	Specifies the name of the instance, hierarchical instance, power domain, or instance group whose objects are to be colored. You can specify an object list, such as `objectName {A,B}`. *Data_type*: string, optional
`-obj_type` *typeList*	Specifies the list of type of objects that are to be colored, such as `BusGuide`, `Module`, `PowerDomain`, and `InstGroup`. *Data_type*: string, optional
`-reset`	Resets the object color. *Data_type*: bool, optional

Example

- The following command colors the `DTMF_INST/TDSP_CORE_INST` instance with color `3`, as specified in the Module Color Preference form:
  ```
  gui_set_obj_color -obj_name DTMF_INST/TDSP_CORE_INST -color_id 3
  ```
- The following command colors the `DTMF_INST/TDSP_CORE_INST` instance and its children with color `3`, as specified in the Module Color Preference form:
  ```
  gui_set_obj_color -obj_name DTMF_INST/TDSP_CORE_INST -color_id 3 -include_children color
  ```

- The following command colors the children of the `DTMF_INST/TDSP_CORE_INST` instance in multiple colors:

  ```
  gui_set_obj_color -obj_name DTMF_INST/TDSP_CORE_INST -include_children multicolor
  ```

Related Information

- `gui_clear_hinst_colors`
- `gui_set_select_hinst_color`
- `write_color_preference`
- `write_hinst_colors`
- Floorplan Menu chapter in the *Menu Reference*
  - *Color Module*

gui_set_power_rail_display

```
gui_set_power_rail_display
[-help]
[-enable_result_browser {true | false}]
[-enable_rlrp {true | false}]
[-enable_voltage_sources {true | false}]
[-filter_max max_value]
[-filter_min min_value]
[-legend {off ne nw se sw}]
[-plot plot_type]
[-range_max max_value]
[-range_min min_value]
[-reset_color_scale {true | false}]
[-filter_color {on | off | color_name}]
[-enable_percentage_range {true | false}]
[-thermal_layer_name thermal_layer]
[-tile_count_x value]
[-tile_count_y value]
[-tile_size_x value]
[-tile_size_y value]
[-customized_range {value1 value2 ... value9}]
[-grid {on | off}]
[-enable_auto_apply {true | false}]
```

Controls the display of the main window widgets, such as legend, result browser, voltage source, filter range, and RLRP path.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `gui_set_power_rail_display` parameter. For a detailed description of the command and all of its parameters, use the man command: `man gui_set_power_rail_display`.
`-customized_range {value1 value2 ... value9}`	

	Specifies the color scale range values for the 8 filters between the minimum and maximum values of the color gradient based (*Basic*) filter. The value list must contain 9 values. The filter ranges for the 8 buckets are evenly divided by default. This parameter can be used to customize the filter ranges. You can either modify the color range using the GUI (Basic button) or the `-customized_range` parameter. **Example**: `gui_set_power_rail_display -plot ir -customized_range { 1 10 12 13 14 15 16 17 49 }` *Data_type*: string, optional
`-enable_auto_apply {true | false}`	
	Specifies to automatically redraw the layout after modifying the color scale range values. This parameter allows you to enable/disable the *Auto Apply for Color Scale* checkbox of the Power Rail Plots form using the Tcl script instead of the GUI option. The default value of this parameter is `false`. *Data_type*: bool, optional
`-enable_percentage_range {true | false}`	
	Specifies to display the color gradients for the following plots by percentage in the GUI: • IR Drop • IVDD • IVDN The percentage value is based on the nominal voltage value of each rail/instance. It displays high percentage voltages in red and low percentage voltages in blue. *Default* : `false` *Data_type*: bool, optional
`-enable_result_browser {true | false}`	
	When set to `true` it enables the result browser. *Default* : `false` *Data_type*: bool, optional

`-enable_rlrp {true | false}`	
	When set to `true` it enables the display of the RLRP path. The rail analysis must include RLRP analysis (`set_rail_analysis_config -enable_rlrp_analysis true`) in order to have the required data for this functionality. *Default* : `false` *Data_type*: bool, optional
`-enable_voltage_sources {true | false}`	
	Displays voltage sources (power pads) using white pixels in the layout canvas. *Default* : `false` *Data_type*: bool, optional
`-filter_color {on | off | color_name}`	
	Allows to set the color of the filtered out resistors specified using the scissor filter range. By default, the filtered resistors are not displayed. If you want to view these resistors, you can use this parameter to enable the visibility of resistors and set desired colors for them. The valid values of this parameter are: • `on` – enables the visibility of the filtered out resistors. • `off` - disables the visibility of the filtered out resistors. • *`color_name`* - enables the visibility of filtered out resistors and set the color of resistors per the specified color name. *Data_type*: string, optional
`-filter_max max_value`	

	Specifies the maximum value of the scissor filter range that will be plotted. You can use the scissor filter range to filter out the resistors and nodes that you do not want to view, and obtain statistics only for the instances in the specified filter range. Optional, but if specified, the `-plot` option is required, and requires both `-filter_min` and `-filter_max` to be specified. *Data_type*: float, optional
`-filter_min` *max_value*	
	Specifies the minimum value of the scissor filter range that will be plotted. Optional, but if specified, the `-plot` option is required, and requires both `-filter_min` and `-filter_max` to be specified. *Data_type*: float, optional
`-grid {on | off}`	
	Allows you to control the grid display when viewing the Voltage Source Current (VC) and Voltage across Package Model (VU) plots. The default value of this parameter is `on`. You can set this parameter to `off` to hide the grid view for better display and easier analysis of the plot data. *Data_type*: string, optional
`-legend {off ne nw se sw}`	
	Displays the filter color and range details. When you set `gui_set_power_rail_display -legend` to one of the location (`ne nw se sw`) , this opens the legend at the specified location on the design layout. Use `gui_set_power_rail_display -legend off` to close the legend. *Default*: `off` *Data_type*: enum, optional
`-plot` *plot_type*	
	Plots the specified plot type. See Plot Types for a description of each of the plot types. *Data_type*: enum, optional
`-range_max` *max_value*	

	Specifies the maximum value of the color scale. Optional, but if specified, the `-plot` option is required, and requires both `-range_min` and `-range_max` to be specified. *Data_type*: float, optional
`-range_min` *min_value*	
	Specifies the minimum value of the color scale. Optional, but if specified, the `-plot` option is required, and requires both `-range_min` and `-range_max` to be specified. *Data_type*: float, optional
`-reset_color_scale {true | false}`	
	Specifies to reset the maximum (`-range_max`) and minimum (`-range_min`) values of the color scale to the default plot data values. For example, when you specify `-plot ir`, the default value for `-range_min` is `0` and `-range_max` is `100`. You can modify the color scale, such as set `-range_min` to `0` and `-range_max` to `20`. To reset the color scale of the IR drop plot to default, you can specify the `-reset_color_scale` parameter. *Data_type*: bool, optional
`-thermal_layer_name` *thermal_layer*	
	Displays the specified layer of the temperature map. **Example:** `set_power_rail_display -thermal_layer_name VIA_7` *Data_type*: string, optional
`-tile_count_x` *value*	

	Specifies the number of tiles in the x direction. This parameter is used to resize the tile power density plot. You can either specify the tiles by count (`tile_count_x` and `tile_count_y`) or specify the tiles by size (`tile_size_x` and `tile_size_y`). When you resize the tile power density (`tpd`) plot in the batch or GUI mode, the software automatically invokes the `report_power` command and resizes the plot. This feature allows you to resize the `tpd` plot without having to restore the power database (`power.db`) again. *Data_type*: int, optional
`-tile_count_y` *value*	
	Specifies the number of tiles in the y direction. This parameter must be specified with the `tile_count_x` parameter. *Data_type*: int, optional
`-tile_size_x` *value*	
	Specifies the horizontal dimension of each tile polygon in *µm*. This parameter is used to resize the tile power density plot. You can either specify the tiles by count (`tile_count_x` and `tile_count_y`) or specify the tiles by size (`tile_size_x` and `tile_size_y`). When you resize the tile power density (`tpd`) plot in the batch or GUI mode, the software automatically invokes the `report_power` command and resizes the plot. This feature allows you to resize the `tpd` plot without having to restore the power database (`power.db`) again. The following is an example of specifying tiles by size: `gui_set_power_rail_display -plot tpd -tile_size_x 100 -tile_size_y 100` *Data_type*: float, optional
`-tile_size_y` *value*	
	Specifies the vertical dimension of each tile polygon in *µm*. This parameter must be specified with the `tile_size_x` parameter. *Data_type*: float, optional

Plot Types

Plot Type	Description
`ac`	Available/feasible capacitance (Farad) Displays the feasible decoupling capacitance that can be added in the area. The feasible capacitance in the area is calculated based on the filler cells placement. The program matches the placed filler cells with the decap cells available in the power-grid library and computes the feasible capacitance. The capacitance of the decap cell is stored in the power-grid library.
`cap`	Ggrid capacitance (Farad) Plots the capacitance of the extracted power-grid network.
`cvi1`	Customized File 1 Plots the user-specified instance voltage file.
`dd`	Decoupling capacitance (decap) density (Farad/um ^{2}) Plots decap density of the design. The decap density is the cell internal decoupling capacitance in the unit area (µm ^{2}).
`dr`	Decoupling capacitance required Reports the decoupling capacitance required to fix dynamic IR-drop violations. This option suggests how much decap needs to be added in a given area to fix the IR-drop violations. The voltage threshold used to determine the IR-drop violations for this analysis is the value specified by the `-threshold` option of the `set_pg_nets` command.
`effr`	Effective resistance (ohm) - instance based Display the effective resistance results for all instances.
`node_reff`	Effective resistance (ohm) - node based Display the effective resistance results for all nodes.
`ipc`	Instance Peak Current Display the peak current of each instance.

`ivdn`	Instance Voltage Drop (Volts) – net-based Plots the net-based instance voltage drop.
`ivdd`	Instance Voltage Drop (Volts) – domain-based Plots the domain-based instance voltage drop. For domain based, at least one power and one ground net must be selected.
`fd`	Filler cell density (1/µm ^{2}) Displays the filler cell density per unit area (µm ^{2}). The plot can be used to analyze the feasibility of adding decap cells by replacing the filler cells and not changing the placement of the logic cells.
`freq`	Frequency domain (Hz) Displays the associated clock frequency of all instances in the design. The instances associated with multiple clock domains are displayed using the fastest clock frequency. This plot can be used to analyze and debug the power distribution in the design.
`ip`	Instance total power (mW) The total power consumed by each instance is displayed. The total power is the sum of the internal, switching and leakage power of the instance. The power data is read from the power database and can be specified using the `-power_db` option or it is queried directly from memory, if power calculation is run during the session.
`ip_i`	Instance internal power (mW) The power consumed by charging and discharging of the internal interconnect and device capacitances of the instance.
`ip_l`	Instance leakage power (mW) The power consumed by devices when not switching.
`ip_s`	Instance switching power (mW) The power consumed by charging and discharging of the interconnect or output load.
`ipd_i`	Specifies internal power density in mW/um ^{2} .

`ipd_l`	Specifies leakage power density in mW/um ^{2} .
`ipd_s`	Specifies switching power density in mW/um ^{2} .
`ipd_t`	Specifies total power density in in mW/um ^{2} .
`ir`	IRdrop or voltage drop (Volts) Displays the IRdrop across each extracted power-grid segment in the design.
`jrms`	Specifies current density as j/jmax based on the Irms current. This plot is only supported for dynamic analysis. A jrms gif file will be saved in the state directory that shows where the Irms based EM violation occurred. This plot type allows you to verify the analysis results of "jrms" using the GUI.
`load`	Loading capacitance (Farad) Displays the output loading capacitance driven by the instance. This plot can be used to analyze and debug the regions with high switching power.
`pi`	Power-switch (gate) current violation: I/Idsat Displays the ratio of current through the power-switch and the saturation current (Idsat) of the power-switch. This plot can be used to analyze and debug whether the current through power-switches exceeds the saturation current characterized for the power-switch devices. The saturation current and on-resistance of the power-switches are characterized and stored inside the power-grid library of the power-switch cell. A ratio of greater than 1 means that the current requirement of the power-gated block can not be met with placed power-switch instance. Either the power-switch needs to be made larger or power-switch instances need to be added.
`ptpd`	Profiling Tile Power Density Displays the tile-based power density plot for the vector profiling flow. You can enable this plot using the `read_power_rail_results -profiling_power_file` *`file`* command. You can also view the power density for the different tile co-ordinates in the *Details* tab of the Result Browser.

`pv`	Voltage drop across power-switch (Volt) Displays the IRdrop across power-switch instances. This plot can be used to analyze and debug the IRdrop inside the power-gated block, e.g. if the IRdrop across power-switches is already high, the IRdrop inside the power-gated block will be much higher. In this case, the power-switch placement will have to be refined in order to resolve the IRdrop problem.
`rc`	Resistor current (Amp) Displays the current across power-grid resistor segments. This plot can be used to analyze and debug how the current flows from the voltage sources into the rest of the design, e.g. high current should flow from pads to the top-level routing level layers and then to the lower level metal layers. This plot can also be used to debug current density violations (`rj`).
`rlrp`	The total resistance of the instance along the least resistive path. Displays the total resistance of the instance along the least resistive path, meaning the resistance of the instance along the path from where it gets the most amount of current. This plot can be used to analyze and debug the power-grid integrity early in the design stage. An instance with high resistance is likely to have high static or dynamic IRdrop depending on the vector that is simulated later in the design cycle. This plot can be also used during decap optimization, since regions with high dynamic IRdrop and high resistance may not be improved upon by adding decaps, because the effectiveness of decoupling capacitance depends on the resistance of the power-grid network. The plot displays the RLRP paths for power (purple color) and ground (white color) nets in different colors for the identification of paths and ease of analysis. You can also highlight the least resistive path of the instance using the GUI and selecting the instance with left mouse button. The *RLRP Path* checkbox should be enabled.

`res`	Grid resistance (ohm) Displays the total resistance of the least resistive path from each node to voltage source. This plot can be used to view high resistance regions of the power-grid. It is useful during IRdrop and decap analysis to identify weak power-grid connectivity. The GUI also allows you to view the resistor-based RLRP path. To view the resistor-based RLRP path: 1. Select the *res* Rail plot, and the *RLRP Path* checkbox. 2. Select a resistor. The GUI highlights the least effective resistance path to the nearest voltage source, and opens the *Resistance Path* form.
`rj`	Resistor current density, J/Jmax. Displays the current density (current per unit area) violation on each power-grid segment using the ratio of calculated current density and maximum allowed current density. A ratio of greater than 1 means that the current density limit of the segment is violated. The current density limits are supplied by the foundry and can be read during analysis using the `set_rail_analysis_mode -em_models` command.
rs	Specifies the plot type for resistor sensitivity.
`sem_avg`	Average Current Density Displays average current limit violations in the design. This plot is enabled when you specify `check_ac_limits -method avg`.
`sem_peak`	Peak Current Density Displays peak current limit violations in the design. This plot is enabled when you specify `check_ac_limits -method peak`.
`sem_rms`	RMS Current Density Displays RMS current limit violations in the design. This plot is enabled when you specify `check_ac_limits -method rms`.
`slack`	Instance slack (Seconds) Displays worst instance slack. The slack data is queried from either the timing database or if the power database is specified using `-power_db` option, then the worst slack data is read from the power database. When using an external TWF, the slack data is read from the TWF.

`tc`	Tap current (Amp) Displays the current consumed by the instance devices or current taps. During rail analysis, it is based on the power consumption of the instance and is distributed inside the cell. The current distribution inside the cell is based on the extracted device netlist stored inside the power-grid library of the cell. The current distribution is generally based on the device width, length, and spice models. This plot can be used to analyze and debug IRdrop inside the macro.
`td`	Transition density (Hz) Displays the worst transition density (activity * frequency) of the instance. This plot can be used to analyze and debug activity distribution and high power density regions.
`thermal`	Instance thermal Displays the instance temperature distribution for the chip. The plot shows tile-based temperature for instances. When you click on an instance, the *Attribute Viewer* window opens and displays the instance temperature in the *Thermal* field.
`unc`	Unconnected segments Displays power-grid segments that are disconnected from the voltage source (power pads). This plot can be used to debug unusually high IRdrop in the region.
`vc`	Voltage source current (Amp) Displays current through the voltage sources (power pads). This plot can be used to analyze the current carrying capacity of the power-pad.
`vu`	Voltage drop across package (Volt) Displays voltage drop across package. If analysis is run with package model attached to the power pads, this plot displays the IRdrop due to package RLCK elements.

Examples

- The following command specifies that `div` data will be plotted without loading design DEF:

```
gui_set_power_rail_display\
-plot div
```

- The following command enables the visibility of the filtered out resistors without changing the color:

```
gui_set_power_rail_display -filter_color on
```

- The following command enables the visibility of the filtered out resistors and also changes the color to red:

```
gui_set_power_rail_display -filter_color red
```

Related Information

- `gui_set_power_rail_layers_nets`
- "Static Power, IRdrop and EM Analysis" in the *Voltus Stylus Common UI User Guide*

gui_set_power_rail_layers_nets

```
set_power_rail_layers_nets
[-help]
[-design_opacity value]
[-rail_analysis_opacity value]
[-nets {{all_power 0/1}{all_ground 0/1}} |
 -enable_nets {names} |
 -disable_nets {names} ]
[-enable_switch_nets {all|net_names} |
 -disable_switch_nets {all|net_names} ]
[-enable_report_layers {all|layer_names} |
 -disable_report_layers {all|layer_names} |
 -layers {{name report0/1 visible0/1}{...}} ]
[-enable_visible_layers {all|layer_names} |
 -disable_visible_layers {all|layer_names} |
 -layers {{name report0/1 visible0/1}{...}} ]
[-enable_visible_signal_em_layers {all|layer_names} |
 -disable_visible_signal_em_layers {all|layer_names} ]
[-show_layer_net_setting_list {true | false}]
[-instance_voltage_method {Worst | Best | Avg | WorstAvg}]
[-instance_voltage_window {1| 0}]
[-via_size {1 | 2 | 3 | 4 | 5 | 6}]
```

Controls the visibility of the layers, the opacity of the rail analysis or design database, and the visibility of power/ground nets.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `gui_set_power_rail_layers_nets` parameter. For a detailed description of the command and all of its parameters, use the man command: `man gui_set_power_rail_layers_nets`.
`-design_opacity value`	

	Specifies the design opacity value for the design database. The opacity range is from 0-100%, the default value is `60`. *Data_type*: float, optional
`-disable_nets {`*names*`}`	
	Hides the specified nets. This parameter is mutually exclusive to the `-enable_nets` and `-nets` parameters. *Data_type*: string, optional
`-disable_report_layers {all|`*layer_names*`}`	
	Hides the report (user-defined layers to be used for processing) layers. You can use `all` to hide all layers, or list the specific layers that are not to be displayed. This parameter is mutually exclusive to the `-enable_report_layers` and `-layers` parameters. *Data_type*: string, optional
`-disable_switch_nets {all|`*net_names*`}`	
	Hides the specified switch nets. You can use `all` to hide all the switch nets, or list the specific switch nets that are not to be displayed. This parameter is mutually exclusive to the `-enable_switch_nets` parameter. *Data_type*: string, optional
`-disable_visible_layers {all|`*layer_names*`}`	
	Specifies which layers should not be made visible when plotting. You can use `all` to hide all layers, or list the specific layers that are not to be displayed. This parameter is mutually exclusive to the `-enable_visible_layers` and `-layers` parameters. *Data_type*: string, optional
`-disable_visible_signal_em_layers {all|`*layer_names*`}`	

	Specifies which signal EM layers should not be made visible when plotting. You can use `all` to hide all layers, or list the specific layers that are not to be displayed. This parameter is mutually exclusive to the `-enable_visible_signal_em_layers` parameter. *Data_type*: string, optional
`-enable_nets {`*names*`}`	
	Displays only the specified nets. This parameter is mutually exclusive to the `-disable_nets` and `-nets` parameters. *Data_type*: string, optional
`-enable_report_layers {all|`*layer_names*`}`	
	Specifies to select the user-defined layers to be used for processing. You can use `all` to select all layers, or list the specific layers that are to be selected. This parameter is mutually exclusive to the `-disable_report_layers` and `-layers` parameters. *Data_type*: string, optional
`-enable_switch_nets {all|`*net_names*`}`	
	Displays the specified switch nets. You can use `all` to display all the switch nets, or list the specific switch nets that are to be displayed. This parameter is mutually exclusive to the `-disable_switch_nets` parameter. *Data_type*: string, optional
`-enable_visible_layers {all|`*layer_names*`}`	
	Specifies which layers should be made visible when plotting. You can use `all` to display all layers, or list the specific layers that are to be made visible. This parameter is mutually exclusive to the `-disable_visible_layers` and `-layers` parameters. *Data_type*: string, optional
`-enable_visible_signal_em_layers {all|`*layer_names*`}`	

	Specifies which signal EM layers should be made visible when plotting. You can use `all` to select all layers, or list the specific layers that are to be selected. This parameter is mutually exclusive to the `-disable_visible_sem_layers` parameter. *Data_type*: string, optional
`-instance_voltage_method {Worst | Best | Avg | WorstAvg}`	
	Allows different methods of computing effective instance voltage during rail analysis. Using this option, you can write out the Best, Worst, or Average effective instance voltage between power and ground nets. The WorstAvg method computes the average instance voltage during each evaluation window defined by the IV Window field, and reports the worst average instance voltage out of all the windows. If IV Method is WorstAvg and IV Window is whole, then this method will report average voltage across all time steps in simulation. Default : `Worst` *Data_type*: enum, optional
`-instance_voltage_window {1| 0}`	
	Controls the EIV plot. EIV is obtained by processing the voltage waveforms and finding the worse-case effective voltage between the power and ground pins during a specific window. There are two options to view EIV plots: • `1` – both switching and timing windows (window EIV) • `0` – worst of the entire rail simulation (worst EIV) *Data_type*: bool, optional
`-layers {{name report0/1 visible0/1}{...}}`	

	Controls the visibility of layers. You can enable or disable the reporting and visibility of specific layer. For example, `-layer {{M1 1 1} {M2 0 0}` means enable reporting and visibility of layer M1, and disable reporting and visibility of layer M2 This parameter is mutually exclusive to the `-disable_report_layers, -disable_visible_layers, -enable_visible_layers,` and `-enable_report_layers` parameters. *Data_type*: string, optional
`-nets {{all_power 0/1}{all_ground 0/1}}`	
	Controls the display of all the power and ground nets. When `all_power` is set to `0`, hides all power nets, and when set to `1`, displays all power nets. When `all_ground` is set to `0`, hides all ground nets, and when set to `1`, displays all ground nets. This parameter is mutually exclusive to the `-enable_nets` and `-disable_nets` parameters. *Data_type*: string, optional
`-rail_analysis_opacity` *value*	
	Specifies the opacity value for the rail analysis output. The opacity range is from 0-100%, the default value is `100`. *Data_type*: float, optional
`-show_layer_net_setting_list {true | false}`	

	Displays the current status (OFF/ON) of all layers and nets in the console. The following is an example of the output:
	`#-------------------------------#`
	`# Layers  Report  Visible #`
	`# RD  ON  ON #`
	`# M0  ON  ON #`
	`.....`
	`#-------------------------------#`
	`# PowerNets  Status #`
	`# VDD  ON #`
	`# VDDm  ON #`
	`#-------------------------------#`
	`# GroundNets Status #`
	`# VSS  ON #`
	`#-------------------------------#`
	Data_type: bool, optional

`-via_size {1 | 2 | 3 | 4 | 5 | 6}`	
	Specifies to modify and control the size of vias in the GUI. The default via size is `1`. The possible via size values are `1`, `2`, `3`, `4`, `5`, and `6`.

Examples

- The following command specifies the layers that will be visible:

```
gui_set_power_rail_layers_nets \
 -enable_visible_layers{RD M1 V12 M2 V23 M3 V34 M4 V45 M5 V56 M6} \
 -enable_nets {vdd vdd_1}
```

- The following commands describe the use model for enabling and disabling switch nets. Here, consider a net `VDD` with three switch nets `VDD_sw0`, `VDD_sw1`, and `VDD_sw2`:
  - The following command hides the switch net `VDD_sw0`:

    ```
    set_power_rail_layers_net -disable_switch_nets {VDD {VDD_sw0}}
    ```

    Displays `VDD_sw1` and `VDD_sw2`.
  - The following command hides all the switch nets:

    ```
    set_power_rail_layers_net -disable_switch_nets {VDD {all}}
    ```
  - The following command displays the switch net `VDD_sw0`:

    ```
    set_power_rail_layers_net -enable_switch_nets {VDD {VDD_sw0}}
    ```
  - The following command displays all the switch nets:

    ```
    set_power_rail_layers_net -enable_switch_nets {VDD {all}}
    ```

Related Information

- `gui_set_power_rail_display`
- "Static Power, IRdrop and EM Analysis" in the *Voltus Stylus Common UI User Guide*

gui_set_select_hinst_color

```
gui_set_select_hinst_color
[-help]
[-color_id colorID]
```

Specifies the color Id for the selected hierarchical instances.

Parameters

`-help`	Prints a brief description that includes type and default information for each `gui_set_select_hinst_color` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man gui_set_select_hinst_color`
`-color_id colorID`	Specifies the color Id to apply to all selected hierarchical instances. The color value for the color id is the same as specified in the Module Color Preference form (accessed by clicking *Floorplan - Edit Floorplan - Color Module*) *Default*: If no Id is specified, the color Id `0` is applied to all the selected hierarchical instances. *Data_type*: int, optional

Example

The following command applies color Id 5 to all the selected hierarchical instances:

```
gui_set_select_hinst_color -color_id 5
```

Related Information

- `gui_clear_hinst_colors`
- `gui_set_obj_color`
- `write_color_preference`
- `write_hinst_colors`
- Floorplan Menu chapter in the *Menu Reference*
    - *Color Module*

gui_set_tool

```
gui_set_tool
[-help]
mode
```

Selects specified tool widget in the main window and enters interactive mode.

For more information on tool widgets, see the "Tool Widgets" section in the The Main Window chapter of the *Innovus Menu Reference*.

Parameters

`-help`	Prints a brief description that includes type and default information for each `gui_set_tool` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man gui_set_tool`
mode {select | move | ruler | util | cut | obstruct | layerBlk | ptnFeed | ptnPinBlk | ptnPinGuide | addWire | addVia | cutWire | stretchWire | addPoly | addBusGuide | p2pRoute}	
	Specifies the interactive mode to enter. The following modes are supported: • `select` - Chooses the Select tool widget in the main window. • `move` - Selects the Move/Resize/Reshape tool widget in the main window. • `ruler` - Selects the Create Ruler tool widget in the main window. • `util` - Selects the Query Area Density tool widget in the main window. • `cut` - Selects the Cut Rectilinear tool widget in the main window. • `obstruct` - Selects the Create Placement Blockage widget in the main window. • `layerBlk` - Selects the Create Routing Blockage tool widget in the main window.

	• `ptnFeed` - Selects the Create Physical Feedthrough tool widget in the main window. • `ptnPinBlk` - Selects the Create Pin Blockage tool widget in the main window. • `ptnPinGuide` - Selects the Create Pin Guide tool widget in the main window. • `addWire` - Selects the Edit Wire tool widget in the main window. • `addVia` - Selects the Add Via tool widget in the main window. • `cutWire` - Selects the Cut Wire tool widget in the main window. • `stretchWire` - Selects the Move/Resize Wire tool widget in the main window. • `addPoly` - Selects the Add Polygon tool widget in the main window. • `addBusGuide` - Selects the Edit Bus Guide tool widget in the main window. • `p2pRoute` - Selects the Point to Point Route tool widget in the main window. *Data_type*: string, required

Example

The following command is used to enter the Create Ruler mode in the design window.

```
gui_set_tool ruler
```

Related Information

- The Main Window chapter of the *Menu Reference*
  - *Tool Widgets*

gui_set_ui

```
gui_set_ui
[-help]
name
[-before name]
[-checked {true | false}]
[-command string]
[-disabled {true | false}]
[-enable_variable string]
[-foreground string]
[-geometry string | -full_screen {true | false}]
[-icon filename]
[-in name]
[-label string]
[-message string]
[-movable {true | false}]
[-newline {true | false}]
[-shortcut string]
[-statusbar string]
[-title string]
[-tooltip string]
[-underline integer]
[-value string ]
[-variable string]
[-visible {true | false}]
[-visible_variable string]
```

Sets the specified property of an element. Use this command to customize the Innovus graphical interface by configuring various interface elements, such as submenus and commands. The properties that can be set for a specific element depends on the `type` of that element. To see the list of properties that can be set for each type, see the "Accepted Properties for Each Type" section below.

For elements of type `menu`, `submenu`, `command`, `check`, `radio`, `separator`, `toolbar`, and `toolbutton`, the `gui_set_ui` command works like the tk command *`widget`*`configure`. For the main window (`main`), it works like the tk command `wm`.

Parameters

`-help`	Prints a brief description that includes type and default information for each `gui_set_ui` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man gui_set_ui`
`name`	Specifies the name of the element whose property is to be set. An error is returned if specified *`name`* does not exist. *Data_type*: string, required
`-before` *`name`*	Sets the specified element before the element identified by *`name`*. *Data_type*: string, optional
`-checked {true | false}`	Checks or unchecks the specified element. **Note:** You can use the `-checked` parameter only for elements of type `check` or `radio`. *Data_type*: bool, optional
`-command` *`string`*	Specifies the Tcl command to be executed when the specified element is invoked. **Note:** You can use the `-command` parameter only if you are adding elements of type `command, check, radio` or `toolbutton`. *Data_type*: string, optional
`-disabled {true | false}`	Disables or enables the specified interface element. **Note:** You can use the `-disabled` parameter only for elements of type `menu, submenu, command, check, radio` or `toolbutton`. The parameter is NOT applicable for elements of type `toolbar` and `separator`. *Data_type*: bool, optional
`-enable_variable` *`string`*	Sets the Tcl global variable for Enable status. *Data_type*: string, optional

`-foreground` *`string`*	Specifies the foreground color of the icon of the specified interface element. **Note:** You can use the `-foreground` parameter only if you are adding an element of type `toolbutton`. *Data_type*: string, optional
`-full_screen {true | false}`	
	Maximizes the main window. This property can be set only for the element `main`. **Note:** This option cannot be used with `-geometry`. *Data_type*: bool, optional
`-geometry` *`string`*	Modifies the geometry of the specified window element. *Data_type*: string, optional
`-icon` *`filename`*	Specifies the file name of the toolbutton icon. *`filename`* can be the full path of the icon file or an icon name supported by Innovus. The icon names used in Innovus can be queried using the `gui_get_ui` command. For list of supported image formats for the icon file, see the "Supported Image Formats" section of the Customizing the User Interface chapter in the *User Guide*. **Note:** The `-icon` parameter is applicable only for elements of type `toolbutton`. *Data_type*: string, optional
`-in` *`name`*	Specifies the name of the parent element. *Data_type*: string, optional
`-label` *`string`*	Specifies the label to be displayed on the element being modified. **Note:** This parameter is NOT applicable for elements of type `separator`. *Data_type*: string, optional

`-message` *string*	Specifies the message to be displayed on the element being modified. **Note:** This parameter is applicable for elements of type `statusbar`. *Data_type*: string, optional
`-movable {true | false}`	Specifies whether or not the element being modified should be movable. **Note:** The `-movable` parameter is applicable only for elements of type `toolbar`. *Data_type*: bool, optional
`-newline {true | false}`	Specifies whether or not the element being modified should be on a new row. **Note:** The `-newline` parameter is applicable only for elements of type `toolbar`. *Data_type*: bool, optional
`-shortcut` *string*	Specifies the shortcut key for a command, check box, radio button, or toolbutton. **Note:** You can use the `-shortcut` parameter only if you are modifying elements of type `command`, `check`, `radio`, or `toolbutton`. *Data_type*: string, optional
`-statusbar` *string*	Sets the statusbar of the specified window element. *Data_type*: string, optional
`-title` *string*	Specifies the window title. *Data_type*: string, optional
`-tooltip` *string*	Specifies the tooltip text for the toolbar or toolbutton you are adding. **Note:** The `-tooltip` parameter is applicable only for elements of type `toolbar` and `toolbutton`. *Data_type*: string, optional

`-underline` *integer*	Specifies the integer index of the character to be underlined in the element label. The index begins with `0` and should have a value less than or equal to the length of the element label. For instance, if you want the second letter of the element label to appear underlined, specify *integer* as `1`. **Note:** The `-underline` parameter is NOT applicable for elements of type `toolbar, separator,` and `toolbutton`. *Data_type*: int, optional
`-value` *stringValue*	Specifies the value to be stored in the associated variable when the specified radio button is selected. **Note:** The `-value` parameter is applicable only for elements of type `radio`. *Data_type*: string, optional
`-variable` *string*	When modifying a check box or radio button, specifies the Tcl global variable for the item. *Data_type*: string, optional
`-visible {true | false}`	Specifies the visibility status of the element being modified. *Data_type*: bool, optional
`-visible_variable` *string*	Sets the Tcl global variable for Visible status. *Data_type*: string, optional

Accepted Properties for Each Type

The accepted properties for each type are listed below.

- For the main window (`main`), the following properties can be set:

  `-geometry {`*widthxheight* `|` *widthxheight+x+y*`}`

  `-title` *title*

  `-close_command` *string*

  `-visible true | false`

- For a status bar (`statusbar`), the following property can be set:

  `-enable_variable` *string*

  `-in` *parentname*

```
-message msg
-visible true | false
-visible_variable string
```

- For elements of type `menu` or `submenu`, the following properties can be set:

```
-enable_variable string
-before nextSiblingName]
-disabled true | false
-in parentName
-label label
-underline index
-visible true | false
-visible_variable string
```

- For elements of type `command`, `radio`, and `check`, the following properties can be set:

```
-disabled true | false
-label label
-underline index
-command commandName
-in parentName
-before nextSiblingName
-enable_variable string
-visible true | false
-visible_variable string
-shortcut string
```

  For `radio` and `check`, the `-checked` and `-variable` properties can also be set.

- For elements of type `toolbutton`, the following properties can be set:

```
-state normal | disabled
-label label
-tooltip toolTipText
-underline index
-foreground colorName
-icon iconName
-command commandName
```

```
-in parentName
-before nextSiblingName
-enable_variable string
-visible true | false
-visible_variable string
-shortcut string
```

Examples

- The following command changes the label of the *Verify* menu in the main window to *NewCheck*:
  ```
  gui_set_ui verify -label NewCheck
  ```
- The following command maximizes the Innovus main window:
  ```
  gui_set_ui main -full_screen
  ```

Related Information

- gui_add_ui
- gui_delete_ui
- gui_get_ui
- gui_find_ui
- Customizing the User Interface chapter in the *User Guide*
  - *Querying and Configuring Interface Elements*

gui_set_visible_bumps

```
gui_set_visible_bumps
[-help]
{-bump_cells bump_cell_name| -all}
```

Displays only the bumps with the specified bump cell names and hides all other bumps.

To redisplay all bumps, specify the `-all` parameter.

Parameters

`-help`	Prints a brief description that includes type and default information for each `gui_set_visible_bumps` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man gui_set_visible_bumps`
`-all`	Displays all bumps. ⚠ By default, all bumps are displayed. When you specify `gui_set_visible_bumps bump_cell_name`, the tool displays only the bumps with the specified bump cell names and hides all other bumps. You can then use `gui_set_visible_bumps -all` to redisplay all bumps.
`-bump_cells names`	Displays only the bumps with the specified bump cell names and hides all other bumps. You can specify multiple bump cell names.

Examples

- The following command displays only the bumps with bump cell as BUMPCELL:
  ```
  gui_set_visible_bumps -bump_cells BUMPCELL
  ```
- The following command displays all bumps:
  ```
  gui_set_visible_bumps -all
  ```

Related Information

- gui_get_visible_bumps

gui_set_visible_net_groups

```
gui_set_visible_net_groups
[-help]
{net_group_names | -all}
```

Displays only the specified net groups and hides all other net groups.

To display all net groups, specify the `-all` parameter.

Parameters

`-help`	Prints a brief description that includes type and default information for each `gui_set_visible_net_groups` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man gui_set_visible_net_groups`
`-all`	Displays all the net groups.
net_group_names	Displays only the specified net groups and hides all other net groups. You can specify multiple net group names.

Examples

- The following command hides all net groups except the specified one:
  ```
  gui_set_visible_net_groups test_ng
  ```

Related Information

- `gui_get_visible_net_groups`

gui_set_visible_nets

```
gui_set_visible_nets
[-help]
{nets | -all}
```

Displays only the specified nets and hides all other nets.

To display all nets, specify the `-all` parameter.

Parameters

`-help`	Prints a brief description that includes type and default information for each `gui_set_visible_nets` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man gui_set_visible_nets`
`-all`	Displays all nets. *Data_type*: bool, optional
nets	Displays only the specified nets and hides all other nets. You can specify multiple net names. *Data_type*: (net)*, optional

Examples

- The following command hides all nets except the specified net:
  ```
  gui_set_visible_nets inst1/lower/net_1
  ```

Related Information

- `gui_get_visible_nets`

gui_show

```
gui_show
[-help]
```

Opens the Innovus main window.

If you run the `gui_show` command when the Innovus main window is already displayed, the tool retains the current zoom level and displays the main window without refreshing it. For example:

```
innovus 23> zoom_box 94 -640 95 -645
innovus 24> gui_show
```

This displays the main window with the design zoomed to the coordinates " `94 -640 95 -645` ".

If you started the Innovus software with the `innovus -no_gui` command, you can use `gui_show` to pop up the main window.

You can run the `gui_hide` command to hide the Innovus main window.

Parameters

`-help`	Prints out the command usage. For a detailed description of the command, use the `man` command: `man gui_show`

Related Information

- `gui_hide`
- The Main Window chapter of the *Menu Reference*
    - *Closing or Hiding the GUI*

gui_show_edge_number

```
gui_show_edge_number [-help]
[-objects string]
[-reset]
```

Displays edge numbers on the sides of the selected or specified objects in the GUI. Currently, you can display edge numbers on partitions and blocks. This feature is useful if you need to indicate the partition or block edge number in your script.

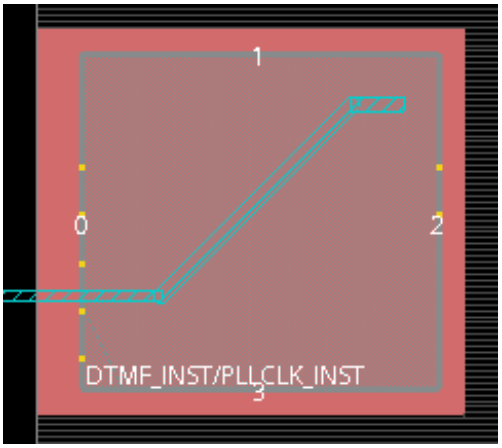
Parameters

`-help`	Prints a brief description that includes type and default information for each `gui_show_edge_number` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man gui_show_edge_number`
`-objects` *`string`*	Specifies the name of the object for which the edge numbers should be displayed. If you want to specify multiple object names, enclose them in braces or double quotation marks. Currently, you can display the edge numbers for partitions and blocks. *Data_type*: string, optional
`-reset`	Clears the edge numbers from the specified or selected objects on the GUI. If you want to clear the edge numbers for all objects, use the `-reset` parameter with `-objects *`. *Data_type*: bool, optional

Examples

- The following command displays the edge numbers on the `DTMF_INST/PLLCLK_INST` block in the GUI:

  ```
  gui_show_edge_number -objects DTMF_INST/PLLCLK_INST
  ```

- The following command clears the edge numbers from all blocks:

  ```
  gui_show_edge_number -objects * -reset
  ```

Related Information

- `gui_select`

gui_show_floorplan_snapshot

```
gui_show_floorplan_snapshot
[-help]
-dir string
[-name string]
[-view string]
```

Opens a window *SnapShot Result* for viewing saved snapshots. Snapshots are selected based on the options: `-dir` and `-name`. The window displays design views controlled by the `-view` option and also shows QoR parameters (WNS,TNS, Placement Overlap, and congestion) saved at the time of creating the snapshot.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `gui_show_floorplan_snapshot` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man gui_show_floorplan_snapshot`.
`-dir` *string*	Specifies the directory of the saved snapshot, when opted alone it will show all snapshots saved in the directory. *Data_type*: string, required
`-name` *string*	Specifies the name of snapshot to be viewed in the *SnapShot Result* window. *Data_type*: string, optional
`-view` *string*	Specifies the view of the design you want to see in *SnapShot Result* window. You can see multiple views by listing them under this option. *Default:* The fplan view is shown in the *SnapShot Result* window. *Data_type*: string, optional

Note: This command will not work in the `-nowin` mode.

Examples

- The following command will show all the snapshots present in the directory `/work/user1`, along with all QoR parameters (WNS, TNS, Placement Overlap, and Congestion). By default only the fplan view will be displayed.

  ```
  gui_show_floorplan_snapshot -dir {/work/user1}
  ```

- The following command will show snapshots with the name `cts` along with all QoR parameters (WNS, TNS, Placement Overlap, and Congestion). By default only the fplan view will be displayed.

  ```
  gui_show_floorplan_snapshot -dir {/work/user1} -name cts
  ```

- The following command will show snapshots with the name `cts` along with all QoR parameters (WNS, TNS, Placement Overlap, and Congestion). Only design views `cts`will be shown as opted under the -view option.

  ```
  gui_show_floorplan_snapshot -dir {/work/user1} -name cts -view cts
  ```

- The following command will show all the snapshots present in the directory `/work/user1`along with all QoR parameters (WNS, TNS, Placement Overlap, and Congestion). All the views will be shown side by side.

  ```
  gui_show_floorplan_snapshot -dir {/work/user1} -view {fplan amoeba place}
  ```

Related Information

- `gui_create_floorplan_snapshot`
- Tools Menu chapter of *Menu Reference*
  - *Create Snapshot*
  - *View Snapshot*

gui_show_user_bundle_nets

```
gui_show_user_bundle_nets
[-help]
[-keep_existing]
-net_bundle bundleName
[-remove]
```

Displays flightlines for the specified net bundle. Bundled nets help you with data flow analysis before and after layout. Net bundles can be created and viewed in floorplan and amoeba views. When you run the `gui_show_user_bundle_nets` command, Innovus switches automatically to the net bundle mode and clears flightlines based on selection.

You can create flight line net bundles using the `gui_create_user_bundle_net` command and delete them using the `gui_delete_user_bundle_net` command.

Note: When selecting objects, you can choose to display flightlines for only bundled nets by using the *Only Show Net Bundle Flight Line* option in the Preferences form. If this option is not selected, the net bundle flightlines are displayed along with other flightlines.

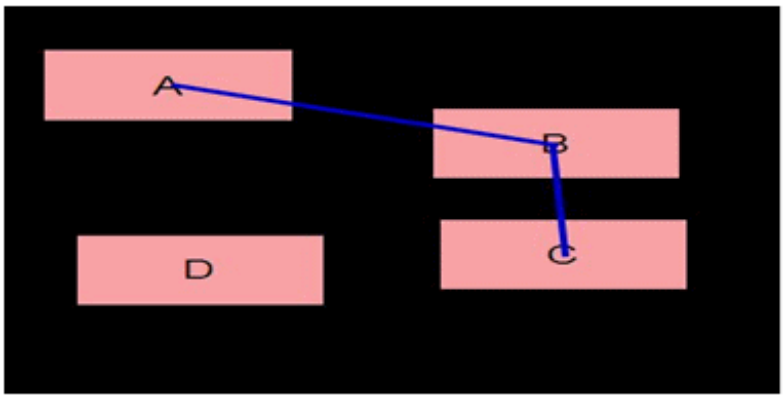
Parameters

`-help`	Prints a brief description that includes type and default information for each gui_show_user_bundle_nets parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man gui_show_user_bundle_nets`
`-keep_existing`	Retains existing net bundle flightlines while displaying the flightline of specified net bundles. If this option is not specified, the command clears existing flightlines before displaying specified net bundle flightlines. *Data_type*: bool, optional
`-net_bundle` *bundleName*	Specifies the name of the net bundle whose flightlines you want to display. You can specify more than one net bundle. *Data_type*: string, required
`-remove`	Clears flightlines of specified net bundle from the display area. If no net bundle name is specified, all flightlines based on net bundles are removed. **Note:** This option only clears the flightlines of the net bundle from the display area. If you want to remove the net bundle itself, use the `gui_delete_user_bundle_net` command. *Data_type*: bool, optional

Examples

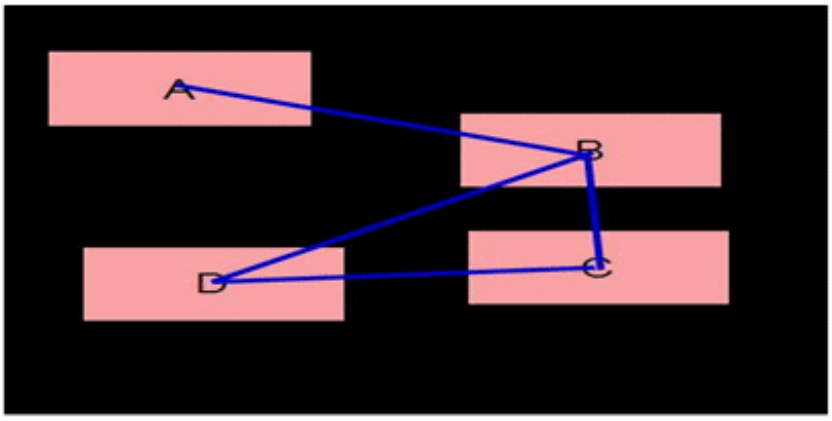
- The following command displays flightlines of the net bundle `bundle_1`. Here, `bundle_1` contains `net_1`, which is just between A and B, and `net_2`, which is just between B and C:

  ```
  gui_show_user_bundle_nets -net_bundle bundle_1
  ```

- The following command displays flightlines of the net bundle `bundle_2` while retaining flightlines of `bundle_1`. Here, `bundle_2` contains `net_3`, which is among B, C, and D:

  ```
  gui_show_user_bundle_nets -net_bundle bundle_2 -keep_existing
  ```

  Note: The same flightlines could also be displayed in a single step by using the following command:

  ```
  gui_show_user_bundle_nets -net_bundle {bundle_1 bundle_2}
  ```

Related Information

- `gui_create_user_bundle_net`
- `gui_delete_user_bundle_net`
- View Menu chapter of the *Menu Reference*
    - *Preferences - Flightline*

gui_trace_flightline

```
gui_trace_flightline
[-help]
-inst instance_name
[-max_inout_number value]
{[-sink  -sink_level value -sink_color string] | [-source  -source_level value -
source_color string]}
```

Traces flightlines for the specified instance from either the output pins (with `-sink`) or the input pins (with `-source`). The flightlines drawn with `gui_trace_flightline` are not dependent on instance selection. These flightlines are cleared only when you use `gui_clear_trace_flightline`.

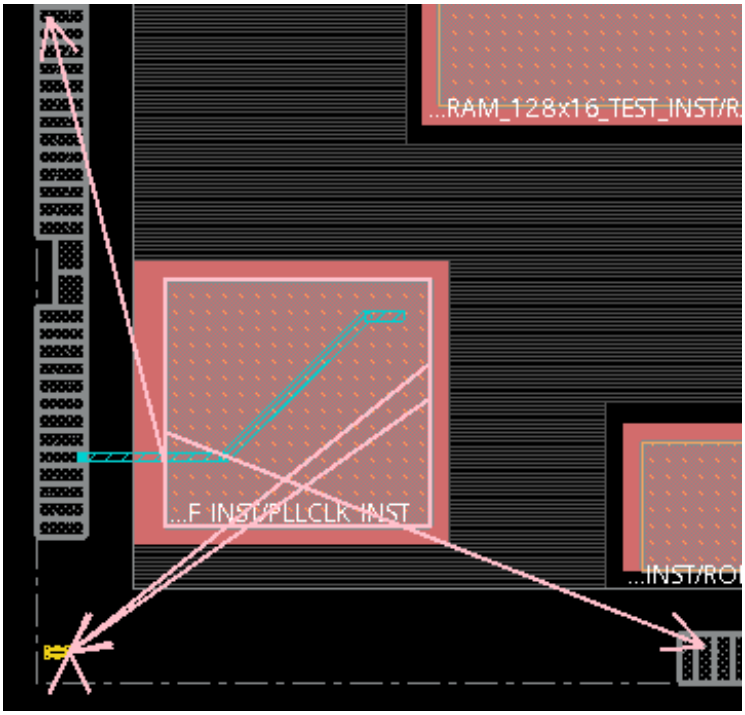
Parameters

`-help`	Prints a brief description of the `gui_trace_flightline` command. For a detailed description of the command, use the `man` command: `man gui_trace_flightline`.
`-inst` *`instance_name`*	Specifies the name of the instance from which the flightline starts. *Data_type*: string, required
`-max_inout_number` *`value`*	Specifies the maximum number of net terms. If the net has more terms than the specified number, the tool skips this net while tracing the flightline. *Default*: 50 *Data_type*: int, optional
`-sink`	Traces flightlines from the output pins of the specified instance. You must also specify `-sink_color` and `-sink_level` with this option. You cannot use this option with any of the `-source*` options. *Data_type*: bool, optional
`-sink_color` *`string`*	Specifies the color of the sink trace. *Data_type*: string, optional
`-sink_level` *`value`*	Specifies the sink trace level. *Data_type*: string, optional
`-source`	Traces flightlines from the input pins of the specified instance. You must also specify `-source_color` and `-source_level` with this option. You cannot use this option with any of the `-sink*` options. *Data_type*: bool, optional
`-source_color` *`string`*	Specifies the color of the source trace. *Data_type*: string, optional
`-source_level` *`value`*	Specifies the source trace level. *Data_type*: string, optional

Examples

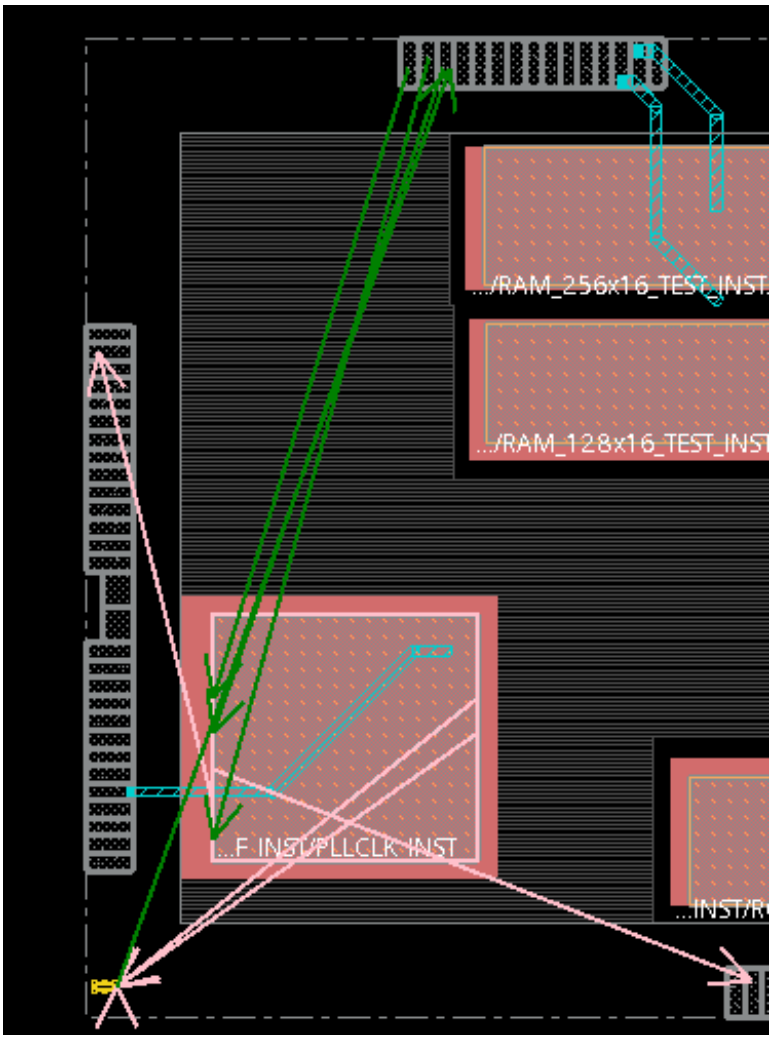
- The following command traces flightlines for the `DTMF_INST/PLLCLK_INST` instance from the output pins for `2` logic levels in the color `pink`:

  ```
  gui_trace_flightline -inst DTMF_INST/PLLCLK_INST -sink -sink_level 2 -sink_color pink
  ```



- The following command traces flightlines for the `DTMF_INST/PLLCLK_INST` instance from the input pins for `2` logic levels in the color `green`:

  ```
  gui_trace_flightline -inst DTMF_INST/PLLCLK_INST -source -source_level 2 -source_color green
  ```



Related Information

- gui_clear_trace_flightline
- gui_report_trace_flightline
- gui_write_flightline

gui_undo

```
gui_undo [-help]
```

Undoes the previous edit made with any bump placement, floorplanning, power planning, or wire edit commands listed below.

- Selected bump placement commands:
  - create_bump
  - delete_bumps
- Selected floorplanning commands:
  - align_selected
  - flip_or_rotate_obj
  - move_obj
  - snap_floorplan
  - snap_floorplan_io
  - space_selected
- Selected power planning commands:
  - add_rings
  - add_stripes
  - update_power_vias
  - route_special
- All wire editor commands:
  - create_fillet
  - convert_special_routes_for_eco
  - delete_fillet
  - delete_routes
  - deselect_routes
  - edit_add_polygon_point

- edit_add_route_point
- edit_add_trim_metal
- edit_create_via
- edit_cut_route
- edit_duplicate_routes
- edit_end_polygon_point
- edit_end_route_point
- edit_merge_routes
- edit_move_routes
- edit_resize_routes
- edit_rotate
- edit_split_routes
- edit_split_wide_routes
- edit_trim_routes
- edit_update_route_layer
- edit_update_route_net
- edit_update_route_rule
- edit_update_route_status
- edit_update_via
- select_routes

You can use the gui_redo command to redo an edit that was reversed with gui_undo.

Parameters

`-help`	Prints a brief description of the `gui_undo` command. For a detailed description of the command, use the `man` command: `man gui_undo`

Related Information

- `gui_redo`
- Edit Menu chapter in the *Menu Reference*
  - *Undo*
  - *Redo*

gui_ungroup_hinst

```
gui_ungroup_hinst
[-help]
[-type {flex_model} | hinst]
[-type {flex_model} [-display_all_constraints ]]
```

Displays the submodules for the specified module guide. The `gui_ungroup_hinst` command is equivalent to the *Ungroup* widget on the toolbar.

If you try to ungroup a module that contains only standard cells and marcos but does not contain any submodules, the tool ignores the ungroup request and issues the following warning:

**WARN: (IMPSYT-3162): Cannot ungroup Module 'DTMF_INST/RAM_128x16_TEST_INST' because it has no child.

Parameters

`-help`	Prints a brief description that includes type and default information for each `gui_ungroup_hinst` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man gui_ungroup_hinst`
`-display_all_constraints`	Displays all constraints, such as fences, guides, regions, and soft guides, regardless of the current hierarchy. When this option is specified, you can view the child constraints. *Data_type*: bool, optional
hinst	Specifies the name of the hierarchical instance (hinst) to be ungrouped. *Data_type*: string, optional
`-type {flex_model}`	Specifies the object type to be ungrouped. At present, only the `flex_model` type is supported. If you specify `-type {flex_model}`, the command ungroups all hierarchical instances or parent hinst down to the flex_module and/or flex_instgroup level. *Data_type*: enum, optional

Examples

- The following command ungroups the specified module and displays its submodules:
  ```
  gui_ungroup_hinst DTMF_INST/TDSP_CORE_INST
  ```
- The following command ungroups all hierarchical instances or parent hinst down to the flex_module and/or flex_instgroup level:
  ```
  gui_ungroup_hinst -type {flex_model}
  ```

Related Information

- `gui_group_hinst`
- The Main Window chapter in *Menu Reference*
  - *Toolbar Widgets*

gui_unregister_edit_callback

```
gui_unregister_edit_callback
[-help]
[-command string]
-edit_type {selection pre-move post-move}
```

Removes the callback function from the specified move action. In GUI, the move action involves the following three steps:

1. Click an object.
2. Move the object to the new location.
3. Click to place the object at the new location and complete the move.

The callback function is implemented based on the move action specified with the `-edit_type` parameter.

Parameters

`-help`	Prints the usage of the command. For a detailed description of the command, use the `man` command: `man gui_unregister_edit_callback`
`-command` *string*	Specifies the name of the callback function. *Data_type*: string, optional
`-edit_type {selection pre-move post-move}`	Specifies the GUI edit function: • `selection` - Implements the callback function when an object is selected (after Step 1). • `pre-move` - Implements the callback function just before the object is placed at the new location (after Step 2, just before you change the coordinates in Step 3). • `post-move` - Implements the callback function after an object is moved (after Step 3, when the tool updates the DB records). *Data_type*: enum, required

Example

Consider the following proc definition:

```
proc move_event_callback {

     set objList [dbGet selected]

     foreach obj $objList {

           set type [dbGet $obj.objType]

           if {$type == "sdp"} {

                custom_snap_sdp $obj

           } elseif {$type == "inst"} {

                set cellName [dbGet $obj.cell.name]

                if {[regexp grf $cellName]} {

                     custom_snap_grf $obj

                }

           }

}

}
```

The following command triggers the `move_event_callback` function on the selected object just before the move is completed:

```
register_gui_edit_callback -edit_type pre_move -command move_event_callback
```

The following command removes the callback to the `move_event_callback` function:

```
gui_unregister_edit_callback -edit_type pre_move -command move_event_callback
```

Related Information

- `gui_register_edit_callback`
- `report_gui_edit_callback`

gui_view_box

```
gui_view_box [-help]
```

Reports the coordinates of the current area being viewed in the GUI.

Parameters

`-help`	Prints out the command usage. For a detailed description of the command, use the `man` command: `man gui_view_box`

Example

The following command reports the coordinates of the current area being viewed in the GUI:

```
innovus 9> gui_view_box

93.367 -646.465 103.294 -638.724
```

Related Information

- `gui_zoom`

gui_write_flightline

```
gui_write_flightline
[-help]
-file file_name
[-net_window]
```

Saves the flightline information of the selected instance to the specified file.

Parameters

`-help`	Prints a brief description of the `gui_write_flightline` command. For a detailed description of the command, use the `man` command: `man gui_write_flightline`.
`-file file_name`	Specifies the name of the file in which the flightline information will be saved. *Data_type*: string, required
`-net_window`	Saves the contents of the Flightline Net Window. *Data_type*: bool, optional

Example

The following command saves the flightline information to a file named `flight1`:

```
gui_write_flightline -file flight1
```

Here's a sample report:

```
#######################################################
# Generated by: Cadence Innovus 18.10-d203_1
# OS: Linux x86_64(Host ID noi-leenap)
# Generated on: Fri Jan 12 09:39:35 2018
# Design: DTMF_CHIP
# Command: gui_write_flightline
#######################################################


DTMF_INST/PLLCLK_INST IOPADS_INST/Pvcopop 1
DTMF_INST/PLLCLK_INST IOPADS_INST/Pvcomop 1
```

```
DTMF_INST/PLLCLK_INST IOPADS_INST/Prefclkip 1
DTMF_INST/PLLCLK_INST IOPADS_INST/Ppllrstip 1
DTMF_INST/PLLCLK_INST IOPADS_INST/Pibiasip 1
```

Related Information

- gui_clear_trace_flightline
- gui_report_trace_flightline
- gui_trace_flightline

gui_write_flow_gifs

```
gui_write_flow_gifs
[-help]
-dir string
[-full_window]
-prefix string
```

Dumps images at various stages of the Innovus Stylus Common UI flow, when executed in the batch mode. The stages include congestion map, cell density map, floorplan view, and colored hierarchy map.

Parameters

`-help`	Prints a brief description that includes type and default information for each `gui_write_flow_gifs` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man gui_write_flow_gifs`
`-dir string`	Specifies the output directory. *Data_type*: string, required
`-full_window`	Maximizes the window. *Data_type*: bool, optional
`-prefix string`	Specifies the file prefix. *Data_type*: string, required

Example

The following command dumps full screen images at the various stages of the Innovus flow to the `TEST` directory:

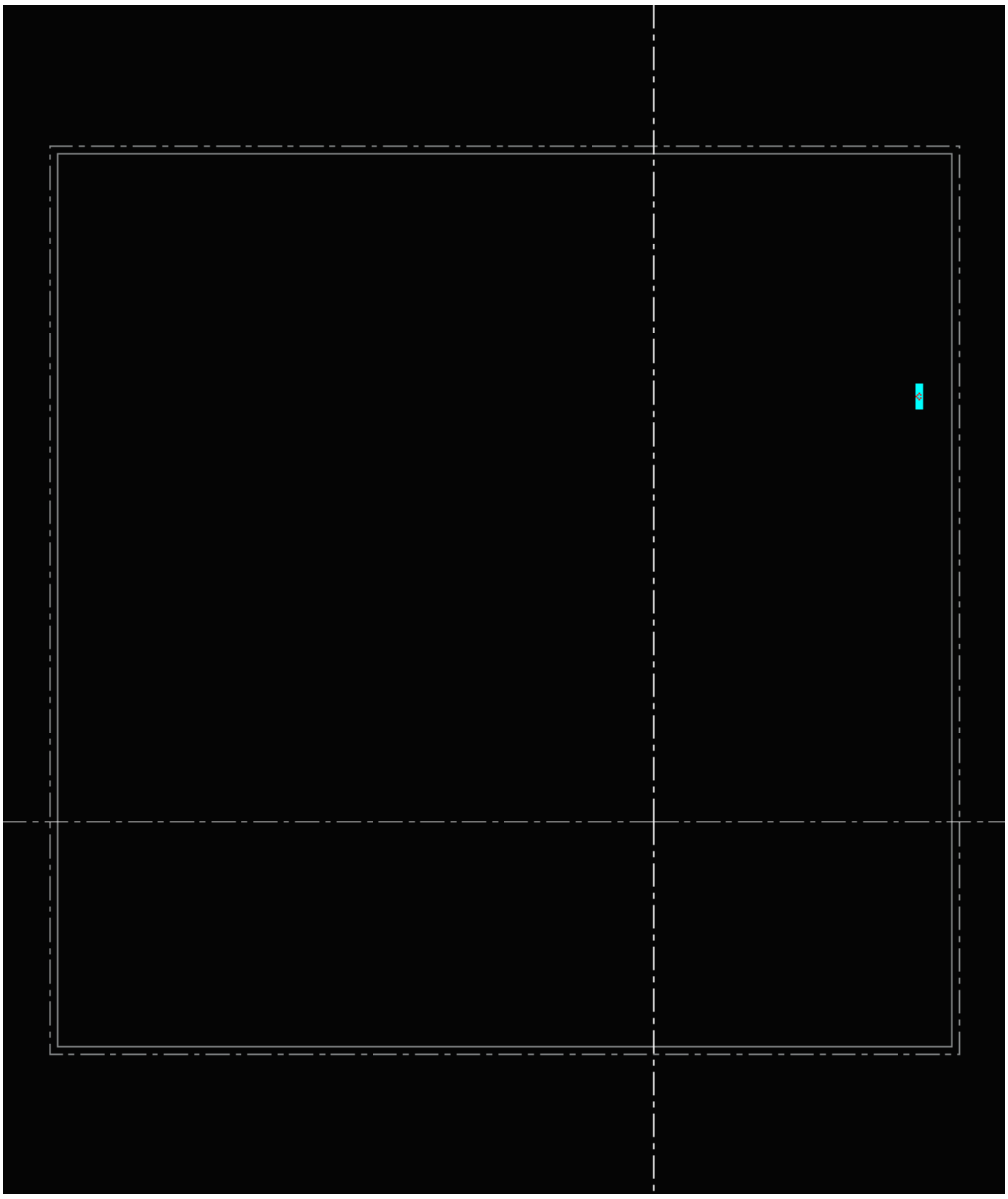
```
gui_write_flow_gifs -dir TEST -prefix [dbGet top.name] -full_window
```

The following images are dumped in the `TEST` directory:

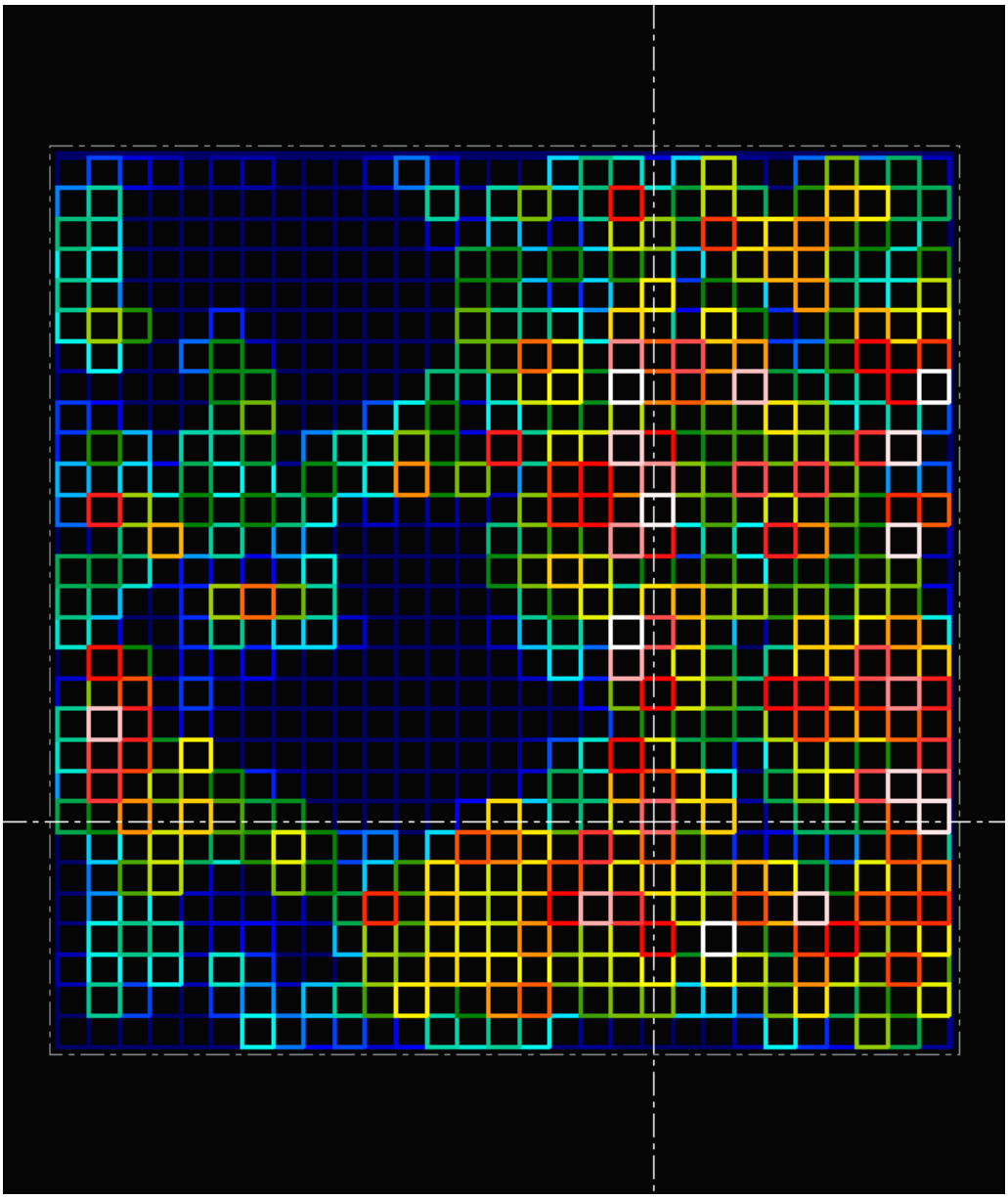
1. `congestion`

2. `density`
3. `floor`
4. `placement`
5. `route_all`
6. `route_C4`
7. `route_C5`
8. `route_C6`
9. `route_M1`
10. `route_M2`
11. `route_M3`
12. `specialroute`
13. `violations`

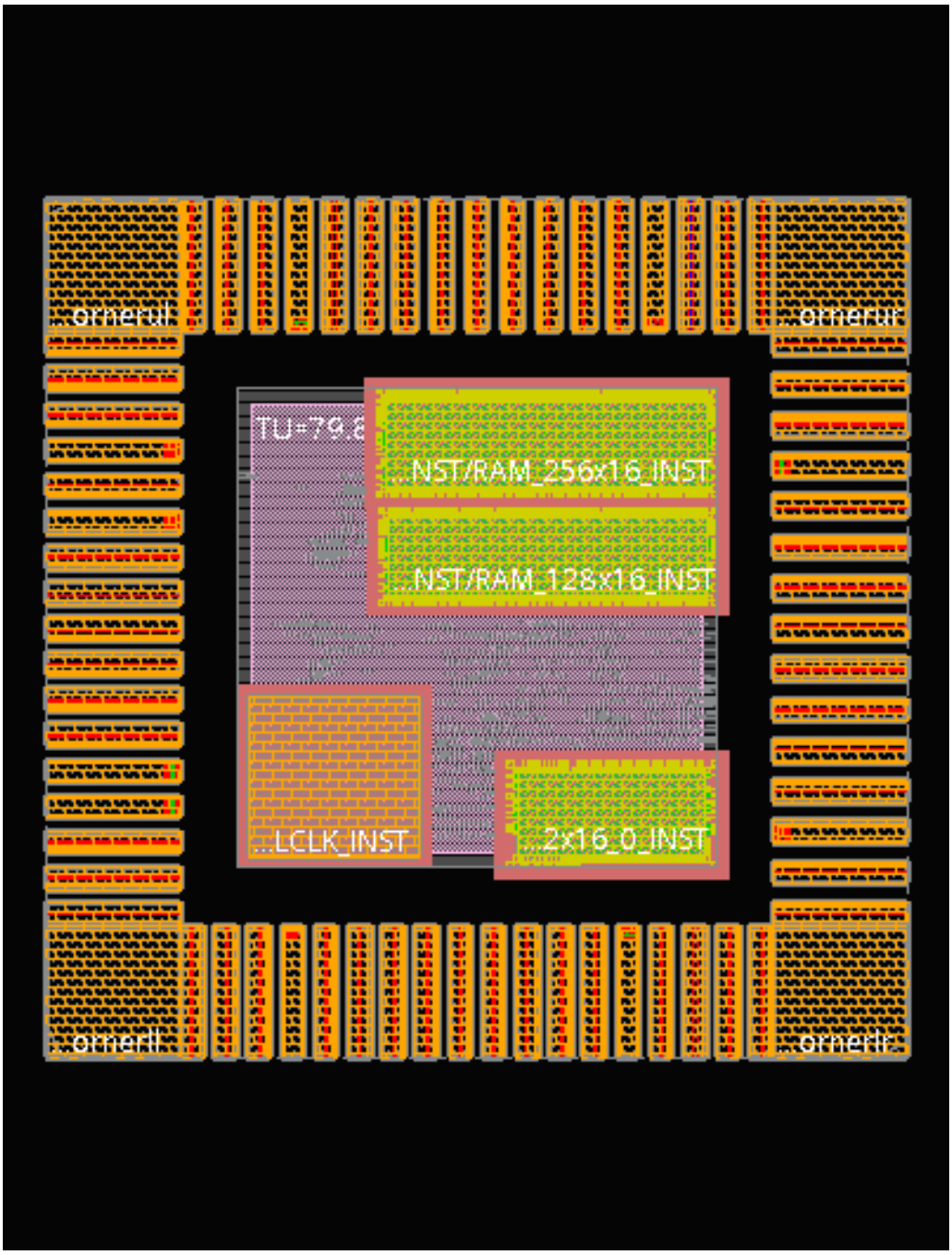
Following is the `congestion` image:



Following is the `density` image:



Following is the floor image:


ornerUL
ornerUR
TU=79.8
...NST/RAM_256x16_INST
...NST/RAM_128x16_INST
...LCLK_INST
...2x16_0_INST
ornerLL
ornerLR

Related Information

- gui_create_floorplan_snapshot
- gui_write_picture
- write_to_gif

gui_write_picture

```
gui_write_picture
[-help]
filename
[-format {bmp gif jpg jpeg png pbm pgm ppm xbm xpm}]
[-width integer_value -height integer_value]
[-window string]
```

Saves a snapshot of the layout or specified window with the specified name in the current directory. You can specify the size as well as the format in which you want to save the snapshot. The command also prints out overlay maps with legends.

You can use this command at any stage of the design flow.

Parameters

`-help`	Prints a brief description that includes type and default information for each `gui_write_picture` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man gui_write_picture`

filename	Specifies the name of the picture file as a string. You can specify the file extension as well.
	
	Note: The file extension you specify does not control the format of the picture. The picture file is dumped in the PNG format by default or the format you explicitly specify with `-format`. For example, the following command saves the `testShot` picture file in the PNG format:
	`innovus 34> gui_write_picture testShot.gif`
	
	current_directory`@file testShot.gif`
	`testShot.gif: PNG image data, 540 x 688, 8-bit/color RGB, non-interlaced`
	
	Data_type: string, required

`-format {bmp gif jpg jpeg png pbm pgm ppm xbm xpm}`	

	Specifies the format of the picture file. *Default*: `png` *Data_type*: enum, optional
`-height` *integer_value*	
	Specifies the height of the picture as an integral value. **Note**: The `-width` parameter is required when `-height` is specified. *Min*: 100 *Max*: 20000 *Data_type*: int, optional
`-width` *integer_value*	
	Specifies the width of the picture as an integral value. **Note**: The `-height` parameter is required when `-width` is specified. *Min*: 100 *Max*: 20000 *Data_type*: int, optional
`-window` *string*	Specifies the ID of the window that you want to dump. You can use the following IDs: • For Layout Viewer: `designview.win` • For Cell Viewer: `.uiCellView.win` If `-window` is not specified, the main layout window is dumped out by default. *Data_type*: string, optional

Examples

- The following command captures the layout window and saves it in the default PNG format with the name *test2* and the dimensions 1000 x 1000 in the current directory:

  ```
  gui_write_picture test2 -width 1000 height 1000
  ```

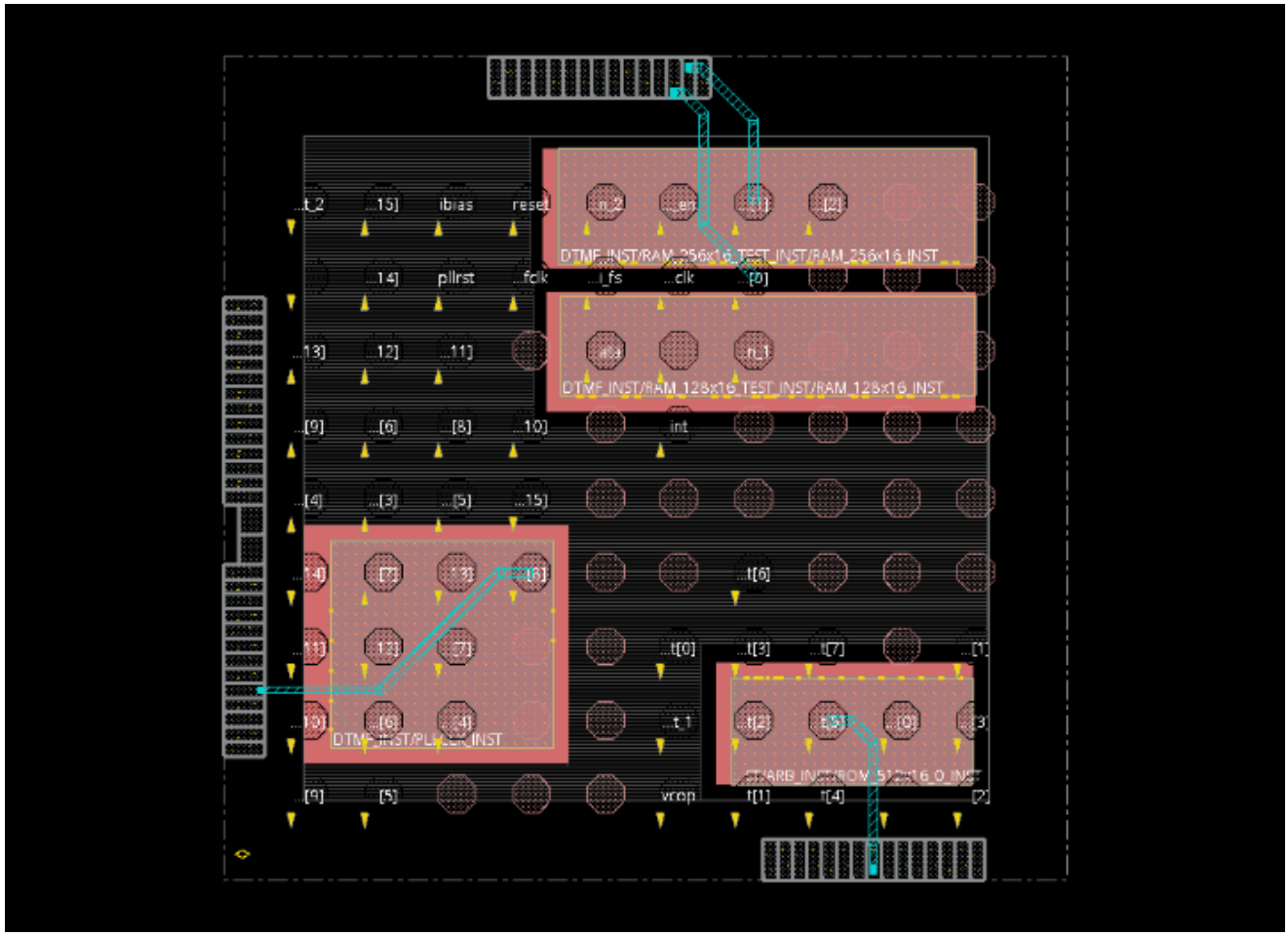
- The following command captures the layout window and saves it in the GIF format with the name *test1.gif* in the current directory:

  ```
  gui_write_picture -format gif test1.gif
  ```

- The following command captures the layout viewer window and saves it in PNG format with the name testShot.png in the current directory:

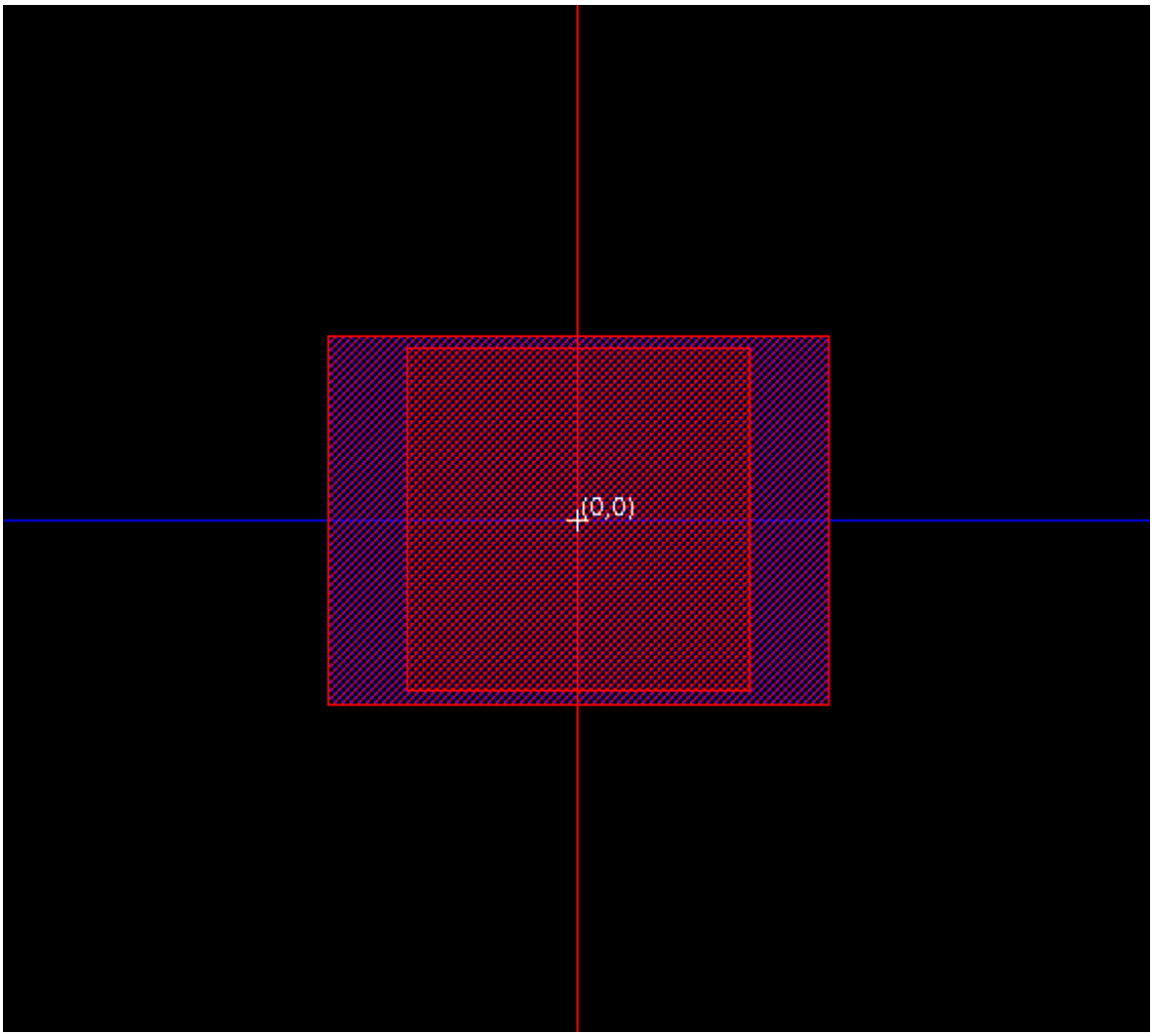
  ```
  gui_write_picture testShot.png -window designview.win
  ```

  Note: You must have the layout viewer open.



- The following command captures the view in the cell viewer window and saves it in PNG format with the name testCell.png in the current directory:

  ```
  gui_write_picture testCell.png -window .uiCellView.win
  ```



Related Information

- gui_create_floorplan_snapshot
- gui_write_flow_gifs
- write_to_gif

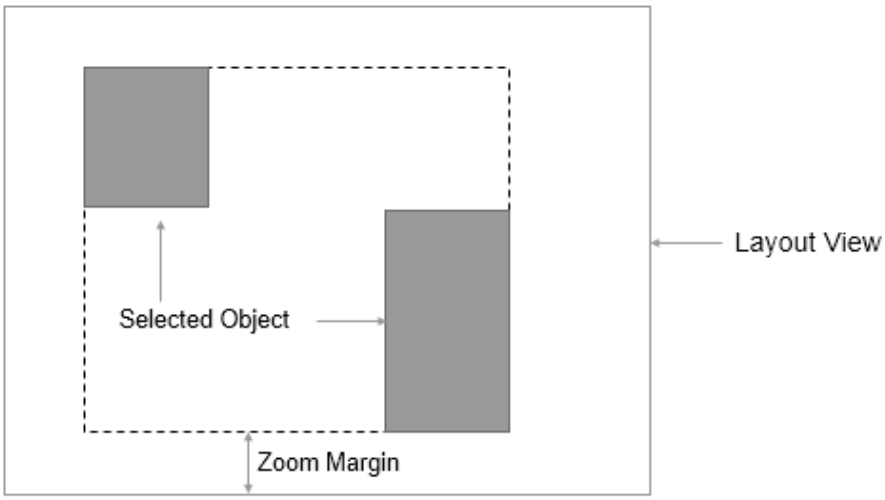
gui_zoom

```
gui_zoom
[-help]
[-to {x y}[-to_radius value]]
[-selected [-margin integer_value]]
{-in  | -out  | -selected  | -to {x y} | -rect {x1 y1 x2 y2}}
```

Zooms into or out of the viewable window as per the specified parameters.

Parameters

`-help`	Prints a brief description that includes type and default information for each `gui_zoom` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man gui_zoom`.
`-in`	Zooms into the viewable window by two times. *Data_type*: bool, optional

`-margin integer_value`	Specifies the zoom margin ratio for the viewable window. The zoom margin is the minimum spacing between the bounding box of the selected object(s) and the layout view.  The minimum value is `0` and the maximum value is `99`. The `gui_zoom -selected` parameter must be specified if you are specifying `-margin`. *Default*: 50 *Data_type*: int, optional
`-out`	Zooms out from the viewable window by two times. *Data_type*: bool, optional
`-rect {x1 y1 x2 y2}`	Zooms in the viewable window to show all of the rectangular area that is defined by the specified coordinates. *Data_type*: rect, optional
`-selected`	Zooms in the viewable window to the area that encloses the selected objects. *Data_type*: bool, optional
`-to {x y}`	Zooms in the viewable window to the point defined by the coordinates, `x` and `y`. *Data_type*: point, optional

`-to_radius value`	Zooms in the viewable window to a radius of the specified value in microns, with center at the point specified by `-to {x y}`. *value* is of type float. *Default*: 150 microns **Note**: The `-to {x y}` parameter must be specified with this parameter. *Data_type*: float, optional

Examples

- The following command zooms in the viewable window to the area enclosed by the lower-left coordinates (`450 531`) and the upper-right coordinates (`550 631`):

  ```
  gui_zoom -rect {450 531 550 631}
  ```

- When the following command is run, the display zooms in to point with x=100 and y=200. The display area is centered at this point and has the default radius of 150 microns:

  ```
  gui_zoom -to {100 200}
  ```

- When the following command is run, the display zooms in to point with x=100 and y=200. The display area is centered at this point and has a radius of 50 microns:

  ```
  gui_zoom -to {100 200} -to_radius 50
  ```

Related Information

- `gui_fit`
- `view_last`
- `view_next`
- View Menu chapter in the *Menu Reference*
  - *Zoom*
- The Main Window chapter in the *Menu Reference*
  - *Select Bar*
  - *View Widgets*

read_power_rail_results

```
read_power_rail_results
[-help]
[-detail_delta_temperature_file file]
[-effective_resistance_file file]
[-instance_delta_temperature_file file]
[-power_db file ]
[-rail_directory dir]
[-tile_delta_temperature_file file]
[-cell_library { library1.cl}]
[-reset]
[-custom_value_inst_file_1 file]
[-nets net_names]
[-instance_voltage_method { best | worst | avg | worstavg }]
[-instance_voltage_window { timing | whole }]
[-instances instancenames]
[-enable_plots plot_names]
[-signal_em_db file]
[-short_file file]
[-esd_directory dir]
[-thermal_file filename.txt]
[-esd_em dir]
[-die_instance_name dieinstname]
[-data_type {rail | all}]
[-physical_db dir_name]
[-profiling_power_file file]
[-top_cell_name topcell_name]
[-tile_thermal_config_file filename]
```

Loading power/rail analysis results into GUI.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `read_power_rail_results` parameter. For a detailed description of the command and all of its parameters, use the man command: `man read_power_rail_results`.
`-cell_library { library1.cl}`	
	Specifies the single cell library that is required to load the state directory. The specified cell library must contain the technology layer information that is required to load the state directory. Therefore, you do not need to load all cell libraries used during analysis. *Data_type*: string, optional
`-custom_value_inst_file_1 file`	
	Specifies the name of the custom value instance file (CVIF). This file can be used to display a user-specified instance voltage file in GUI. The format of the CVIF file is: *value1 instancename1* *value2 instancename2* .... where, *value1* and *value2* are voltage values mapping to the specified instance. You can use the `cvil` argument of the `gui_set_power_rail_display -plot` command parameter to view the EIV map in GUI. *Data_type*: string, optional
`-data_type {rail | all}`	
	Controls the type of data to be loaded in the GUI. The possible arguments of this parameter are: • `rail` - Loads only the rail analysis results. • `all` - Loads both the power and rail analysis results. This is the default value. *Data_type*: enum, optional
`-detail_delta_temperature_file file`	

	Specifies the name of the detailed wire-based delta temperature file from self-heating analysis. *Data_type*: string, optional
`-die_instance_name` *dieinstname*	
	Specifies the name of the die whose results you need to view. This parameter is only applicable to the multi-die analysis flow (`set_rail_analysis_mode -die_mode multi-die`). The `-die_instance_name` parameter must be specified with the `-rail_directory` parameter (multi-die rail directory). *Data_type*: string, optional
`-effective_resistance_file` *file*	
	Specifies the result file from effective resistance analysis (only instance-based result is supported). When the rail analysis plot type `effr` is selected and a result file is specified, the layout shows effective resistance analysis result for each instance. **Note**: Currently, this feature only supports loading one net at a time. *Data_type*: string, optional
`-enable_plots` *plot_names*	
	Specifies the list of power or rail plots names that are to be loaded. *Data_type*: string, optional
`-esd_directory` *dir*	
	Specifies the ESD analysis directory. *Data_type*: string, optional
`-esd_em` *dir*	

	Specifies the name of the directory where the current density/EM results generated by a previous ESD analysis run reside. This parameter allows you to selectively load the required EM result directory if there are multiple ESD EM result directories under the `ESD` directory. You must either load the rail directory (`-rail_directory`) before running this command, or specify the rail directory with the `-esd_em` parameter. *Data_type*: string, optional
`-instances` *instancenames*	
	Specifies a list of instances to load in the GUI. This parameter loads only those partitions that include the specified instances. The `-instances` parameter allows you to analyze parts of the designs, thereby reducing the loading time and memory footprint. The `-instances` parameter is applicable to the Voltus-XP flow only.
`-instance_delta_temperature_file` *file*	
	Specifies the name of the instance-based delta temperature file from self-heating analysis. This is the device instance temperature. *Data_type*: string, optional
`-instance_voltage_method { best | worst | avg | worstavg }`	
	Loads the instance voltage data for the specified EIV computing method. If this parameter is not specified (default), the instance voltage results for all the four methods are loaded. *Data_type*: string, optional
`-instance_voltage_window { timing | whole }`	
	Loads the instance voltage data for the selected EIV window. If this parameter is not specified (default), the instance voltage results for both the timing and switching windows, and the entire rail simulation are loaded. *Data_type*: string, optional
`-nets` *net_names*	

	Loads only the specified nets in the domain-based state directory to the GUI. If this parameter is not specified with the `-rail_directory` parameter, all the nets will be loaded. *Data_type*: string, optional
`-power_db` *file*	
	Specifies the power database file. This file was created by using the `power_write_db` attribute. *Data_type*: string, optional
`-physical_db` *dir_name*	
	Specifies the name of the hierarchical design directory to be loaded to the GUI along with the rail analysis results. *Data_type*: string, optional
`-profiling_power_file` *file*	
	Specifies to display the tile-based power density results generated from the vector profiling flow. *Data_type*: string, optional
`-reset`	
	Frees the loaded analysis state directory or power database from the memory. *Data_type*: bool, optional
`-rail_directory` *dir*	
	Specifies the rail analysis directory. You can load all the domain results by selecting the automatically generated analysis directory under the output directory (`<domain>_<temp>_<dynamic/avg>_#`), or load individual nets by going one level further and selecting the net directory. *Data_type*: string, optional
`-short_file` *file*	

	Specifies the path to the short violations report (`voltus_short.report`) . This file is generated using the `set_rail_analysis_config -report_shorts true` command parameter, and is located in the work directory. *Data_type*: string, optional
`-signal_em_db` *file*	
	Specifies to read the signal electromigration data from the specified database file generated by the `check_ac_limits -report_db` command parameter. *Data_type*: string, optional
`-thermal_file` *filename*`.txt`	
	Specifies to load the tile-based temperature map file generated by Sigrity Celsius to be used for both power calculation and IR drop analysis. The following is an example of reading the thermal map file: `read_power_rail_results -rail_directory output/static_rail/core_25C_avg_1/ -thermal_file output/power_map_by_layer.txt gui_set_power_rail_display -plot thermal` *Data_type*: string, optional
`-tile_delta_temperature_file` *file*	
	Specifies the name of the tile-based delta temperature file of self-heating analysis. This is the worst instance temperature among all wire segments in a tile. This is also layer-based. The FEOL or cell layer is used to show the device instance temperature. *Data_type*: string, optional
`-tile_thermal_config_file` *filename*	

	Specifies to load the configuration file that contains the mapping between the hierarchical levels with corresponding layer-based temperature map files generated by Celsius (system-level thermal analysis tool). When specified, you can view tile-based temperature distribution for each layer of the chip.
	
	The following is the format and example of the configuration file:
	<hierarchy> <block name> <X_tile_no,Y_tile_no> <LEF/DEF Flip/Rotation> <temperature map>
	`TOP DIE 0,0 N /thermal_data/DIE_TemperatureMap`
	`TOP/IP1 IP1 25,50 N /thermal_data/IP1_TemperatureMap`
	`TOP/IP1/B1 B1 50,125 N /thermal_data/B1_TemperatureMap`
	`TOP/IP1/B2 B2 200,125 S /thermal_data/B2_TemperatureMap`
	
	This is a mandatory parameter for the `tile_thermal` plot when loading hierarchical temperature maps.
	
	Data_type: string, optional

`-top_cell_name` *topcell_name*	
	Specifies the name of the top module or top cell name to be loaded. This parameter must be specified with the `-physical_db` parameter. *Data_type*: string, optional

Examples

- The following command specifies the appropriate files needed for viewing the instance voltage and power data:

```
read_power_rail_results\
    -rail_directory analysis/domain_25C_avg_1 \
    -power_db powermeter.db \
```

- The following command specifies to load the VDD and VSS nets for the worst and best EIV methods during the entire simulation:

```
read_power_rail_results -rail_directory dynamic_rail/core_25C_dynamic_1 \
    -nets { VDD VSS } -instance_voltage_window { whole } -instance_voltage_method
{ worst best }
```

- The following command specifies to load only the rail directory:

```
read_power_rail_results -rail_directory dynamic_rail/core_25C_dynamic_1 -data_type
rail
```

- The following command specifies to load the hierarchical database and the rail analysis results to the GUI:

```
read_power_rail_results -rail_directory ./dynamic_rail/latest/ -physical_db
/dma/Dynamic/saveDB -top_cell_name dma_mac
```

- The following command specifies to display the tile-based power density results in the GUI:

```
read_power_rail_results -profiling_power_file
./vectorProfileResults/profile.rpt.density
```

Related Information

- report_power_rail_results
- "Static Power, IRdrop and EM Analysis" in the *Voltus Stylus Common UI User Guide*

read_workspace

```
read_workspace
[-help]
[-dir directory]
-name workspaceName
```

Loads the named workspace from the specified directory. A workspace contains customized main window settings including:

- Size and placement of window
- Specific orientation of toolbars
- Docking status
- Visibility settings

You can use the `read_workspace` command at any stage in the design flow to load a previously saved workspace. However, if you want Innovus to load a specific workspace automatically on starting, you need to first set that workspace as default using the `set_default_workspace` command.

Parameters

`-help`	Prints a brief description that includes type and default information for each `read_workspace` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man read_workspace`
`-dir` *`directory`*	Specifies the directory from which the workspace is to be loaded. The default search order to access directories for loading workspaces is as below: • Current working directory • User home directory The `-dir` parameter is useful if you have workspaces with the same name in both the current working directory and your home directory. If you do not specify `-dir`, the software loads the *`workspaceName`* workspace from the current working directory. *Data_type*: string, optional
`-name` *`workspaceName`*	Specifies the name of the workspace to be loaded. *Data_type*: string, required

Example

The following example loads the `MyWorkspace` workspace from the current working directory.

```
read_workspace -name MyWorkspace
```

Related Information

- `delete_workspace`
- `write_workspace`
- `set_default_workspace`
- Windows Menu chapter of the *Menu Reference*
  - *Workspaces*

report_gui_edit_callback

```
report_gui_edit_callback
[-help]
[-edit_type {selection pre-move post-move}]
[-file string]
```

Reports the callback functions added to the specified move action. In GUI, the move action involves the following three steps:

1. Click an object.
2. Move the object to the new location.
3. Click to place the object at the new location and complete the move.

The callback function is implemented based on the move action specified with the `-edit_type` parameter.

Note that the `report_gui_edit_callback` command does not report the default callback functions.

Parameters

`-help`	Prints the usage of the command. For a detailed description of the command, use the `man` command: `man report_gui_edit_callback`
`-edit_type {selection pre-move post-move}`	Specifies the GUI edit function: • `selection` - Implements the callback function when an object is selected (after Step 1). • `pre-move` - Implements the callback function just before the object is placed at the new location (after Step 2, just before you change the coordinates in Step 3). • `post-move` - Implements the callback function after an object is moved (after Step 3, when the tool updates the DB records). *Data_type*: enum, optional
`-file` *string*	Specifies the name of the file. *Data_type*: string, optional

Examples

Consider the following proc definitions:

```
proc move_callback_1 {
     ...
}
proc move_callback_2 {
     ...
}
proc select_callback_1 {
     ...
}
```

- The following commands add the specified callback functions to the selected object based on the selection and on the post-move operations, as specified below:

```
gui_register_edit_callback -edit_type post-move -command move_callback_1

gui_register_edit_callback -edit_type post-move -command move_callback_2

gui_register_edit_callback -edit_type selection -command select_callback_1
```

- The following command reports all the callback functions on the console:

```
> report_gui_edit_callback

################################
# GUI Callback Function Report
################################
Post Move:
move_callback_1
move_callback_2

Selection:
select_callback_1
```

- The following command prints the callback functions on the post-move operation to the file `cb_move`:

  ```
  report_gui_edit_callback -edit_type post-move -file cb_move
  ```

  The contents of the file is as follows:

  ```
  ##########################################################################

  #  Generated by:      Cadence Innovus 18.10-d133_1

  #  OS:                Linux x86_64(Host ID sjfib212)

  #  Generated on:      Sun Oct  8 19:51:35 2017

  #  Design:            DTMF_CHIP

  #  Command:           report_gui_edit_callback -edit_type post-move -file cb_move

  ##########################################################################

  Post Move:

  move_callback_1

  move_callback_2
  ```

Related Information

- gui_register_edit_callback
- gui_unregister_edit_callback

report_power_rail_results

```
report_power_rail_results
[-help]
[-append]
[-enable_filter_summary]
[-file_name file ]
[-filter_max max_value]
[-filter_min min_value]
[-ignore_limit_bound]
[-layers {All, layer_names } ]
[-limit N ]
[-nets {all | all_pwr | all_gnd | net_names}]
[-plot plot_type]
[-range_max max_value]
[-range_min min_value]
[-region x1 y1 x2 y2]
[-rlrp_inst {instance_name}]
```

Generates RLRP, region-based, and rail/power/capacitance reports.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `report_power_rail_results` parameter. For a detailed description of the command and all of its parameters, use the man command: `man report_power_rail_results`.
`-append`	Appends instance reports to the current report file.
`-enable_filter_summary`	
	Displays a summary of the filtered data in the histogram given in the *Summary* tab of the Result Browser form.

`-file_name` *file*	
	Specifies the name of the text based violation report file to be created. The default file name is: • for region-based report: `region_x1_y1_x2_y2.rpt` • for RLRP report: `rlrp_instname.rpt` • for plot-based report: `power_rail_`*plot*`.rpt` *Data_type*: string, optional
`-filter_max` *max_value*	
	Specifies the maximum value to include in the report. Use this option to further filter out the data within the range specified. This is used to obtain statistic of filter range. Optional, but if specified, the `-plot` option is required, and requires both `-filter_min` and `-filter_max` to be specified. *Data_type*: float, optional
`-filter_min` *max_value*	
	Specifies the minimum value to include in the report. Use this option to further filter out the data within the range specified. This is used to obtain statistic of filter range. Optional, but if specified, the `-plot` option is required, and requires both `-filter_min` and `-filter_max` to be specified. *Data_type*: float, optional
`-ignore_limit_bound`	
	Specifies to ignore the maximum upper limit of `10000`, and instead use the higher value given with the `-limit` parameter for specifying the number of data points to be printed in the power/rail report. *Data_type*: bool, optional
`-layers {all,` *layer_names* `}`	

	Specifies the layers to be reported. You can use `all` for reporting all layers, or list the specific layers that are to be reported. If this parameter is specified, you must specify the `-plot` parameter. *Data_type*: string, optional
`-limit` *N*	
	Specifies the number of data points printed in the report. If this parameter is specified, you must specify the `-plot` parameter. *Default* : first `1000` data points *Data_type*: float, optional
`-nets {all | all_pwr | all_gnd |` *net_names*`}`	
	Specifies the nets to be used for analysis reporting. You can use `all` for reporting all nets, `all_pwr` for reporting all power nets, `all_gnd` for reporting all ground nets, or list the specific nets that are to be reported. If this parameter is specified, you must specify the `-plot` parameter. This parameter also allows you to specify the switched and always-on nets to be used for analysis reporting. The following is the use model to generate a report for the specified switched net: `report_power_rail_results -plot ir -nets {`*switchedNetName*`}` The following is the use model to generate a report for the specified always-on net, including its switched nets: `report_power_rail_results -plot ir -nets {`*alwaysOnNetName*`}` **`-all`** You must specify `-all` to include all switched nets associated with the specified always-on net. *Data_type*: string, optional
`-plot` *plot_type*	
	Plots the specified plot type. See Plot Types for a description of each of the plot types. *Data_type*: enum, optional
`-range_max` *max_value*	

	Specifies the maximum value to report. Optional, but if specified, the `-plot` option is required, and requires both `-range_min` and `-range_max` to be specified. *Data_type*: float, optional
`-range_min` *min_value*	
	Specifies the minimum value to report. Optional, but if specified, the `-plot` option is required, and requires both `-range_min` and `-range_max` to be specified. *Data_type*: float, optional
`-region` *x1 y1 x2 y2*	
	Measures tap current and capacitance information in the specified region defined by x1, y1, and x2 y2 coordinates, and prints results in the log file. *Data_type*: rect, optional
`-rlrp_inst` *instance_name*	
	Reports the RLRP path for the specified instance. The default file name of the RLRP path report is `<`*rlrp_instname*`.rpt>`. You can modify the file name using the `-file_name` parameter. *Data_type*: string, optional

Plot Types

Plot Type	Description

`ac`	Available/feasible capacitance (Farad) Displays the feasible decoupling capacitance that can be added in the area. The feasible capacitance in the area is calculated based on the filler cells placement. The program matches the placed filler cells with the decap cells available in the power-grid library and computes the feasible capacitance. The capacitance of the decap cell is stored in the power-grid library.
`activity`	Displays the worst activity for the output and inout pins of every instance.
`cap`	Grid capacitance (Farad) Plots the capacitance of the extracted power-grid network.
`dd`	Decoupling capacitance (decap) density (Farad/um ^{2}) Plots decap density of the design. The decap density is the cell internal decoupling capacitance in the unit area (µm ^{2}).
`dr`	Decoupling capacitance required Reports the decoupling capacitance required to fix dynamic IR-drop violations. This option suggests how much decap needs to be added in a given area to fix the IR-drop violations. The voltage threshold used to determine the IR-drop violations for this analysis is the value specified by the `-threshold` option of the `set_pg_nets` command.
`ivdn`	Instance Voltage Drop (Volts) – net-based Plots the net-based instance voltage drop.
`ivdd`	Instance Voltage Drop (Volts) – domain-based Plots the domain-based instance voltage drop. For domain based, at least one power and one ground net must be selected.
`fd`	Filler cell density (1/µm ^{2}) Displays the filler cell density per unit area (µm ^{2}). The plot can be used to analyze the feasibility of adding decap cells by replacing the filler cells and not changing the placement of the logic cells.

`freq`	Frequency domain (Hz) Displays the associated clock frequency of all instances in the design. The instances associated with multiple clock domains are displayed using the fastest clock frequency. This plot can be used to analyze and debug the power distribution in the design.
`ip`	Instance total power (mW) The total power consumed by each instance is displayed. The total power is the sum of the internal, switching and leakage power of the instance. The power data is read from the power database and can be specified using the `-power_db` option or it is queried directly from memory, if power calculation is run during the session.
`ip_i`	Instance internal power (mW) The power consumed by charging and discharging of the internal interconnect and device capacitances of the instance.
`ip_l`	Instance leakage power (mW) The power consumed by devices when not switching.
`ip_s`	Instance switching power (mW) The power consumed by charging and discharging of the interconnect or output load.
`ipd_i`	Specifies internal power density in mW/um ^{2} .
`ipd_l`	Specifies leakage power density in mW/um ^{2} .
`ipd_s`	Specifies switching power density in mW/um ^{2} .
`ipd_t`	Specifies total power density in in mW/um ^{2} .
`ip_tr`	Displays toggle rates for all instances. The toggle rate for an instance is calculated as the average of toggle rates of all outputs of the instance. This plot allows you to determine the design area that toggles more than the others.

`ir`	IRdrop or voltage drop (Volts) Displays the IRdrop across each extracted power-grid segment in the design.
`jrms`	Specifies current density as j/jmax based on the Irms current. This plot is only supported for dynamic analysis. A jrms gif file will be saved in the state directory that shows where the Irms based EM violation occurred. This plot type allows you to verify the analysis results of "jrms" using the GUI.
`load`	Loading capacitance (Farad) Displays the output loading capacitance driven by the instance. This plot can be used to analyze and debug the regions with high switching power.
`pi`	Power-switch (gate) current violation: I/Idsat Displays the ratio of current through the power-switch and the saturation current (Idsat) of the power-switch. This plot can be used to analyze and debug whether the current through power-switches exceeds the saturation current characterized for the power-switch devices. The saturation current and on-resistance of the power-switches are characterized and stored inside the power-grid library of the power-switch cell. A ratio of greater than 1 means that the current requirement of the power-gated block can not be met with placed power-switch instance. Either the power-switch needs to be made larger or power-switch instances need to be added.
`pv`	Voltage drop across power-switch (Volt) Displays the IRdrop across power-switch instances. This plot can be used to analyze and debug the IRdrop inside the power-gated block, e.g. if the IRdrop across power-switches is already high, the IRdrop inside the power-gated block will be much higher. In this case, the power-switch placement will have to be refined in order to resolve the IRdrop problem.

`rc`	Resistor current (Amp) Displays the current across power-grid resistor segments. This plot can be used to analyze and debug how the current flows from the voltage sources into the rest of the design, e.g. high current should flow from pads to the top-level routing level layers and then to the lower level metal layers. This plot can also be used to debug current density violations (`rj`).
`rlrp`	The total resistance of the instance along the least resistive path. Displays the total resistance of the instance along the least resistive path, meaning the resistance of the instance along the path from where it gets the most amount of current. This plot can be used to analyze and debug the power-grid integrity early in the design stage. An instance with high resistance is likely to have high static or dynamic IRdrop depending on the vector that is simulated later in the design cycle. This plot can be also used during decap optimization, since regions with high dynamic IRdrop and high resistance may not be improved upon by adding decaps, because the effectiveness of decoupling capacitance depends on the resistance of the power-grid network. You can also highlight the least resistive path of the instance using the GUI and selecting the instance with left mouse button. The *RLRP Path* checkbox should be enabled.
`res`	Displays the total resistance of the least resistive path from each node to voltage source. This plot can be used to view high resistance regions of the power-grid. It is useful during IRdrop and decap analysis to identify weak power-grid connectivity.
`rj`	Resistor current density, J/Jmax. Displays the current density (current per unit area) violation on each power-grid segment using the ratio of calculated current density and maximum allowed current density. A ratio of greater than 1 means that the current density limit of the segment is violated. The current density limits are supplied by the foundry and can be read during analysis using the `set_rail_analysis_mode -em_models` command.
rs	Specifies the plot type for resistor sensitivity.

`slack`	Instance slack (Seconds) Displays worst instance slack. The slack data is queried from either the timing database or if the power database is specified using `-power_db` option, then the worst slack data is read from the power database. When using an external TWF, the slack data is read from the TWF.
`tc`	Tap current (Amp) Displays the current consumed by the instance devices or current taps. During rail analysis, it is based on the power consumption of the instance and is distributed inside the cell. The current distribution inside the cell is based on the extracted device netlist stored inside the power-grid library of the cell. The current distribution is generally based on the device width, length, and spice models. This plot can be used to analyze and debug IRdrop inside the macro.
`td`	Transition density (Hz) Displays the worst transition density (activity * frequency) of the instance. This plot can be used to analyze and debug activity distribution and high power density regions.
`tpd`	Displays power density map with user-specified grid size.
`unc`	Unconnected segments Displays power-grid segments that are disconnected from the voltage source (power pads). This plot can be used to debug unusually high IRdrop in the region.
`vc`	Voltage source current (Amp) Displays current through the voltage sources (power pads). This plot can be used to analyze the current carrying capacity of the power-pad.
`vu`	Voltage drop across package (Volt) Displays voltage drop across package. If analysis is run with package model attached to the power pads, this plot displays the IRdrop due to package RLCK elements.

Examples

- The following command will report the worst 100 IR segments on Metal 1 of the VDD net:
  ```
  report_power_rail_results -plot ir -layers M1 -limit 100 -nets VDD
  ```
- The following command will report the IR drop of the VDD_ring switched net:
  ```
  report_power_rail_results -plot ir -net {VDD_ring}
  ```
- The following command will report the IR drop of the VDD_AO always-on net, including its switched nets:
  ```
  report_power_rail_results -plot ir -net {VDD_AO} -all
  ```
- The following command filters the data such that it does not display the data points below 200 in the Result Browser histogram:
  ```
  report_power_rail_results -plot ivdd -filter_min 200 -enable_filter_summary
  ```

Related Information

- `read_power_rail_results`
- "Static Power, IRdrop and EM Analysis" in the *Voltus Stylus Common UI User Guide*

set_default_workspace

```
set_default_workspace
[-help]
[-dir directory]
-name workspaceName
```

Sets specified workspace as the default. The default workspace is loaded automatically when the software starts. So if you want Innovus to load a workspace automatically at startup, you need to first save it as default.

Parameters

`-help`	Prints a brief description that includes type and default information for each `set_default_workspace` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man set_default_workspace`
`-dir` *`directory`*	Specifies the location of the workspace to be set as default. *Data_type*: string, optional
`-name` *`workspaceName`*	Specifies the name of the workspace to be set as default. *Data_type*: string, required

Example

The following example sets the `TestWorkspace` workspace in the current working directory as default.

```
set_default_workspace -name TestWorkspace
```

Related Information

- `delete_workspace`
- `read_workspace`
- `write_workspace`
- Windows Menu chapter of the *Menu Reference*
    - *Workspaces*

view_last

```
view_last [-help]
```

Displays the last view window or the previous view. This command provides the same function as the Zoom Previous icon (bindkey *w*).

Parameters

`-help`	Prints a brief description of the `view_last` command. For a detailed description of the command, use the `man` command: `man view_last`.

Related Information

- `gui_zoom`
- `view_next`
- The Main Window chapter in the Menu Reference

view_next

`view_next [-help]`

Displays the next view window. This command provides the same function as the *Next* toolbar widget. You can use the `view_next` command or the *Next* toolbar widget to switch from the current display to the next location in the *Go To Saved Area* list or the next zoom level in the zoom sequence saved in the design memory. The equivalent bindkey is `Y`. When you press the `Y` bindkey or execute the `view_next` command repeatedly, the tool cycles through all saved views (saved areas and zoom levels).

Parameters

`-help`	Prints a brief description of the `view_next` command. For a detailed description of the command, use the `man` command: `man view_next`.

Related Information

- `gui_zoom`
- `view_last`
- The Main Window chapter in the Menu Reference

write_color_preference

```
write_color_preference
[-help]
[-dir <string>]
-file_name <string>
[-include_custom]
```

Writes color preferences for the Innovus main display as specified in the Color Preferences form.

Parameters

`-help`	Prints a brief description that includes type and default information for each `write_color_preference` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man write_color_preference`
`-dir` *string*	Specifies the directory path. By default, the color preference file is saved in the current directory. *Data_type*: string, optional
`-file_name` *string*	Specifies the name of the color preference file. *Data_type*: string, required
`-include_custom`	Includes the settings in the Customize Tab form within the Color Preferences form. *Data_type*: bool, optional

Related Information

- `set_preference`
- The Main Window chapter of the Menu Reference
  - *Color Preferences*
- View Menu chapter of the Menu Reference
  - *All Colors*

write_hinst_colors

`write_hinst_colors [-help] out_file`

Saves the current color setting for a hierarchical instance from the Module Color Preference form to a file. The Module Color Preference form is accessed by clicking *Floorplan - Edit Floorplan - Color Module*.

Parameters

`-help`	Prints a brief description that includes type and default information for each `write_hinst_colors` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man write_hinst_colors`.
`out_file`	Specifies the name of the file that stores 30 color settings starting with the `set_layer_preference` command. This is the same behavior as the *Save* button on the Module Color Preference form. *Data_type*: string, required

Example

The following command saves the current color setting for a hierarchical instance from the Module Color Preference form to a file named `hinstColorFile`:

```
write_hinst_colors hinstColorFile
```

Related Information

- `gui_clear_hinst_colors`
- `gui_set_obj_color`
- `gui_set_select_hinst_color`
- `write_color_preference`
- Floorplan Menu chapter in the *Menu Reference*
    - *Color Module*

write_to_gif

`write_to_gif [-help] out_file`

Saves a snapshot of the current screen to a GIF file with the specified name in the current directory. You can use this command at any stage of the design flow.

Parameters

`-help`	Prints a brief description that includes type and default information for each `write_to_gif` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man write_to_gif`.
`out_file`	Specifies the name of the output file as a string. *Data_type*: string, required

Example

The following example captures the current screen and saves it to *`test1.gif`* in the current directory.

```
write_to_gif test1.gif
```

Related Information

- `gui_create_floorplan_snapshot`
- `gui_write_picture`
- `gui_write_flow_gifs`
- Tools Menu chapter of *Menu Reference*
  - *Write To GIF File*

write_workspace

```
write_workspace
[-help]
[-dir directory]
-name workspaceName
```

Saves the named workspace to the specified directory. You can then load the workspace whenever you want from the specified directory by using the `read_workspace` command. A workspace contains customized main window settings including:

- Size and placement of window
- Specific orientation of toolbars
- Docking status
- Visibility settings

Parameters

`-help`	Prints a brief description that includes type and default information for each `write_workspace` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man write_workspace`
`-dir directory`	Specifies the directory in which the workspace is to be saved. *Default*: Current working directory *Data_type*: string, optional
`-name workspaceName`	Specifies the name of the workspace to be saved. *Data_type*: string, required

Example

The following example saves the current window settings to the `TestWorkspace` in the current working directory.

```
write_workspace -name TestWorkspace
```

Related Information

- delete_workspace
- read_workspace
- set_default_workspace
- Windows Menu chapter of the *Menu Reference*
    - *Workspaces*

15

ILM and FlexILM Commands and Attributes

- commit_module_model
- create_ilm_from_files
- flatten_ilm
- get_ilm_type
- ilm Category Attributes
- opt_cloud_design
- place_cloud_design
- read_ilm
- report_ilm_status
- reset_ilm
- set_ilm_type
- unflatten_ilm
- update_partition
- write_ilm
- read_ilm_eco_db
- write_ilm_eco_db

commit_module_model

```
commit_module_model
[-help]
[-copy_mmmc_latencies]
[-merged_mmmc_map_file mapFiles]
[-merged_mmmc_sdc_files mergedSdcFiles]
[-mmmc_file fullChipSdc]
[-mmmc_module_model_tag modelTag]
[-propagate_clocks]
[-cpf_file filename | -1801 filename]
```

Commits module models.

Parameters

`-1801 filename`	Specifies IEEE 1801 file. *Data_type*: string, optional

`-copy_mmmc_latencies`	Copies clock latencies of all analysis views at the top-level design and applies them after assembling the design:
	When this option is specified, the command will do following after assembling the design:
	`foreach view [all_analysis_views] {`
	`update_analysis_view -name $view -latency_file $latency_map($view)`
	`}`
	Where `latency_map` has clock latency files for all views.

`-cpf_file filename`	Specifies the name of the full chip CPF file to be used. *Data_type*: string, optional

`-help`	Outputs a brief description that includes the type and default information for each `commit_module_model` parameter. For a detailed description of the command and all of its parameters, use the man command: `man commit_module_model`
`-merged_mmmc_map_file` *mapFiles*	Specifies merge hierarchical mmmc map files. *Data_type*: string, optional
`-merged_mmmc_sdc_files` *mergedSdcFiles*	Specifies SDC files for different dies. *Data_type*: string, optional
`-mmmc_file` *fullChipSdc*	Specifies the name of the full chip MMMC view file. *Data_type*: string, optional
`-mmmc_module_model_tag` *modelTag*	Specifies MMMC module model tag. *Data_type*: string, optional

`-propagate_clocks`	Changes clock ideal to propagated clock. When this option is specified, you can run the following commands:
	`set_interactive_constraint_modes [all_constraint_modes -active]`
	`set_propagated_clock [all_clocks]`

Related Information

- `set_interactive_constraint_modes`
- `set_propagated_clock`
- `update_analysis_view`

create_ilm_from_files

```
create_ilm_from_files
[-help]
[-cell block_name]
[-incremental]
-model_type {timing | cts | si}
[-overwrite]
[-si_timing_window_file xtwf_file.spef]
{-out_dir directory_name | -oa_cellview lib cell view}
[-verilog_file cell_name.v [-cell_prefix cell_prefix -uniquify_netlist]]
[-sdc_file file_name -timing_view view_name]
[-spef_file spef_file.spef -rc_corner RC_corner_name]
[-opt_stage stage]
```

Creates an ILM directory from existing data. It can only create non-merged models. This command is useful in situations where an ILM is to be created through external inputs or using data from third party models into a format that could be recognized by Innovus.

If the data files already exist in the specified directory, the tool stops and generates an error message. You can overwrite existing files by specifying `-overwrite`. You can add files to the directory by specifying `-incremental`.

The command also creates a tcl script (cell_name_ILM.tcl) that you can source in your command file. The script contains the following command:

```
read_ilm -cell cell_name -out_dir dir_name
```

Parameters

`-cell` *block_name*	Specifies the name of the block. *Data_type*: string, required
`-cell_prefix` *cell_prefix*	Specifies the cell prefix. *Data_type*: string, optional
`-oa_cellview` *lib cell view*	Specifies the library name, cell name, and view name in the OpenAccess database format. *Data_type*: string, optional

`-opt_stage` *stage*	Specifies the optimization stage, preCTS or postCTS. *Data_type*: string, optional
`-out_dir` *dir_name*	Stores the ILM data in the specified directory: the absolute directory path or the path relative to the current directory. *Data_type*: string, optional
`-incremental`	Lets you add files in addition to the existing files to the directory. *Data_type*: bool, optional
`-model_type {timing | cts | si}`	Specifies the type of ILM model, i.e timing, cts or si,. *Data_type*: enum, required
`-help`	Outputs a brief description that includes the type and default information for each `create_ilm_from_files` parameter. For a detailed description of the command and all of its parameters, use the man command: man `create_ilm_from_files`.
`-overwrite`	Overwrites existing files in the directory. *Data_type*: bool, optional
`-rc_corner` *RC_corner_name*	Specifies the RC corner name for MMMC analysis. It identifies the specified corner to be used with -spef_file. *Data_type*: string, optional
`-sdc_file` *file_name*	Specifies an existing .sdc file that you want to include in the directory. It identifies the sdc file to be used with the specified view. *Data_type*: string, optional

`-spef_file` *`spef_name`*`.spef`	Specifies an existing .spef file that you want to include in the directory. It identifies the spef file to be used with the specified corner. *Data_type*: string, optional
`-timing_view` *`view_name`*	Specifies the view name for the ILM module. It identifies the specified view to be used with `-sdc`. *Data_type*: string, optional
`-uniquify_netlist`	Uniquifies the input Verilog netlist. This ensures that there are no errors when ILMs are being loaded. Using this parameter, each sub-module for an ILM has a unique name, which is composed of the time stamp and its original name. For example, in top_sm WedJun31138212009$data_path, top_sm specifies the top-level design, WedJun31138212009 specifies the time stamp, and $data_path specifies the original module name. *Default*: `Off` *Data_type*: bool, optional
`-verilog_file` *`netlist_name`*`.v`	Specifies the verilog file you want to include in the ILM data directory. *Data_type*: string, optional
`-si_timing_window_file` *`xtwf_file.spef`*	Specifies the .xtwf file that you want to include in the directory. *Data_type*: string, optional

Examples

- The following commands create two module directories:

  ```
  create_ilm_from_files -cell blk1 -out_dir A -model_type timing -verilog_file V1.v
  -spef_file 1.spef -rc_corner corner1 -sdc_file sdcfile1 -timing_view view1


  create_ilm_from_files -cell blk2 -out_dir A -model_type si -verilog_file V2.v -
  spef_file 2.spef -rc_corner corner2 -sdc_file sdcfile2 -timing_view view2
  ```

- The command creates two tcl files in the current directory. The `blk1_ILM.tcl` file contains the following command:

  ```
  read_ilm -out_dir A -cell blk1
  ```

  The `blk2_ILM.tcl` file contains the following command:

  ```
  read_ilm -out_dir A -cell blk2
  ```

Related Information

- `read_ilm`

flatten_ilm

```
flatten_ilm
[-help]
```

Flattens the interface logic models (ILMs) at the top level so that the entire design can be analyzed at the top. When you use this command, the software switches from blackbox view to ILM view.

If you do not run `flatten_ilm`, the ILM netlist will not be visible to the Innovus software. You can switch between the ILM view and the blackbox view using the `flatten_ilm` and `unflatten_ilm` commands.

All super commands such as `time_design`, `place_design`, and `opt_design` flatten the ILMs internally and hence no explicit `flatten_ilm` is required. However, you should use this command for other timing related commands. For example, for the `report_timing` command, you need to run `flatten_ilm` before reporting timing. It would not work in the unflattened mode.

Example:

```
flatten_ilm

report_timing
```

Parameters

`-help`	Outputs a brief description that includes the usage of the `flatten_ilm` command.

Related Information

- `opt_design`
- `place_design`
- `report_timing`
- `time_design`
- `unflatten_ilm`
- Top-level Timing Closure Methodologies chapter of the *Innovus Stylus Common UI User Guide*.

get_ilm_type

```
get_ilm_type
[-help]
[-model]
```

Displays the model type set using `set_ilm_type`.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `get_ilm_type` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man get_ilm_type`
`-model`	Outputs the model type. *Data_type*: bool, optional

Examples

- The following example displays the current setting for the `-model` parameter:

  ```
  get_ilm_type -model
  ```

  The software displays the following information:

  ```
  timing
  ```

Related Information

- `set_ilm_type`

ilm Category Attributes

The root attributes in the ilm category are:

- ilm_filter_internal_path
- ilm_keep_async
- ilm_keep_flatten
- ilm_keep_high_fanout_ports
- ilm_keep_inst_in_sdc
- ilm_keep_loopback
- ilm_max_num_insts
- ilm_max_num_registers
- ilm_slack_driven

To set the value of any root attribute, use the `set_db` command:

```
set_db attribute_name value
```

To get the current value of any root attribute, use the `get_db` command:

```
get_db attribute_name
```

To get the current value of all attributes in the `ilm` category, use the following command:

```
get_db -category ilm
```

ilm_filter_internal_path

```
ilm_filter_internal_path {true|false}
```

Default: `false`

Data_type: bool, read/write

When set to true, group paths are created for internal paths with slack adjustment of 9999. Therefore, all internal paths are filtered out from the timing report.

Related Command

- `write_ilm`

ilm_keep_async

```
ilm_keep_async {true|false}
```

Default: `false`

Data_type: bool, read/write

Keeps paths from asynchronous input ports when ILMs are created by `write_ilm`. For example, paths from an input port to the reset pin of interface sequential instances, but not to the internal sequential instances.

Related Command

- `write_ilm`

ilm_keep_flatten

```
ilm_keep_flatten {true|false}
```

Default: `true`

Data_type: bool, read/write

Makes flow work under the flattenIlm mode

ilm_keep_high_fanout_ports

```
ilm_keep_high_fanout_ports {true|false}
```

Default: `true`

Data_type: bool, read/write

Keeps high-fanout input port paths when ILMs are created by `write_ilm`.

Related Commands

- `write_ilm`

ilm_keep_inst_in_sdc

`ilm_keep_inst_in_sdc {true|false}`

Default: `false`

Data_type: bool, read/write

Keeps instances which have constraints on it.

Related Command

- `write_ilm`

ilm_keep_loopback

`ilm_keep_loopback {true|false}`

Default: `false`

Data_type: bool, read/write

Keeps loop back paths when ILMs are created by `write_ilm`.

When set to `true`, then any path touching a net used for the ILM model is also included. This is done for better model accuracy, so that the actual capacitance on an output port could affect the delay on a net which feeds an internal path that is normally not included in the ILM model. This also degrades the reduction ratio (`#ILM_insts/#all_inst`) for the block.

Related Commands

- `write_ilm`

ilm_max_num_insts

`ilm_max_num_insts` *`maxInstances`*

Default: `1000`

Data_type: int, read/write

Specifies the maximum number of instances.

ilm_max_num_registers

```
ilm_max_num_registers maxRegisters
```

Default: `40`

Data_type: int, read/write

Specifies the maximum number of registers.

ilm_slack_driven

```
ilm_slack_driven {true|false}
```

Default: `false`

Data_type: bool, read/write

Specifies slack driven ILM generation.

opt_cloud_design

```
opt_cloud_design
[-help]
{-pre_cts -post_cts}
[-block_type {all flexIlm}]
[-cloud_dir directory]
[-cloud_type {sdc lib reverseIlm}]
[-out_dir directory]
[-prefix prefix]
[-partition_list list_of_partitions | -area {x1 y1 x2 y2} | -top_only  | -block_only ]
```

Provides the ability to optimize chip-level design. It supports three different optimization methods:

- **ReverseILM**: It is a model that represents the design of a particular partition block plus the pushed-down top-level clock network and I/O logics path that made up the “cloud”. With reverseILM technology, tool can make the logic changes inside the partition block but logics that are outside the block design are read-only. The behavior is reverse or opposite to ILM.
- **Color SDC**: It generates SDC for each master/clones, then define a new mode (named with color) to include all the SDCs for master/clones.
- **Rainbow cloud**: It generates new lib and new modes for each master/clones to include more accurate information for optimization

⚠ This command is a limited-access feature in this release. It is enabled by a variable specified using the `set_limited_access_feature` command. To use this feature, contact your Cadence representative to explain your usage requirements, and make sure this feature meets your needs before deploying it widely.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `opt_cloud_design` parameter. For a detailed description of the command and all of its parameters, use the man command: `man opt_cloud_design`

`{-pre_cts -post_cts}`	Specifies the opt_design stage. *Data_type*: enum, required
`-area {x1 y1 x2 y2}`	Specifies area coordinates for area optimization. *Data_type*: string, optional
`-block_only`	Optimizes at the block-level only. *Data_type*: bool, optional
`-block_type {all flexIlm}`	Specifies the block model type. By default, full block-level netlist will be used for the top-level optimization. Otherwise, the flexILM model will be used. *Default*: `all` *Data_type*: enum, optional
`-cloud_dir` *directory*	Specifies the directory to save the cloud opt data. *Data_type*: string, optional
`-cloud_type {sdc lib reverseIlm}`	Specifies the cloud model type. Use sdc, lib, or reverseIlm for color SDC, rainbow cloud, or reverseIlm optimization techniques, respectively. *Default*: `lib` *Data_type*: enum, optional
`-out_dir` *directory*	Specifies the directory to save the report file. *Data_type*:
`-prefix` *prefix*	Specifies the prefix for the report file. *Data_type*: string, optional
`-partition_list` *list_of_partitions*	Specifies the list of partition names that will be optimized at their block-level designs before optimizing the top-level design. *Data_type*: string, optional
`-top_only`	Optimizes at the top-level only. *Data_type*: bool, optional

Related Information

- place_cloud_design

place_cloud_design

```
place_cloud_design
[-help]
[-dir working_directory]
[-no_pre_place_opt]
```

Runs Hierarchical Timing Closure Flow with the reverseILM placement.

The Hierarchical Timing Closure Flow:

- Honors predefined partition fence and module definition
- Uses reverseILM technology to enable pin-less and budget-less flow for the block placement
- Supports distributed placement for quick TAT

⚠ This command is a limited-access feature in this release. It is enabled by a variable specified using the `set_limited_access_feature` command. To use this feature, contact your Cadence representative to explain your usage requirements, and make sure this feature meets your needs before deploying it widely.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `place_cloud_design` parameter. For a detailed description of the command and all of its parameters, use the man command: `man place_cloud_design`
`-dir working_directory`	Specifies the working directory. *Data_type*: string, optional
`-no_pre_place_opt`	Disables the pre-place optimization. *Data_type*: bool, optional

Related Information

- opt_cloud_design

read_ilm

```
read_ilm
[-help]
{[-cell cellName [-directory dirName]] | -oa_cellview lib cell view}
```

Specifies ILM data directory for the specified block. You can run `read_ilm` multiple times in the same session. Each time you specify `read_ilm`, the software overwrites the previous setting for the block. If master/clones exist in the design, the cell name will have the name of the master partition.

Note: You can use this command only if the ILMs are unflattened (`unflatten_ilm`) or the design is in black box view. You cannot change ILM settings in flattened or ILM view.

The syntax is order-dependent. You must specify `-cell` before you specify `-directory`.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `read_ilm` parameter. For a detailed description of the command and all of its parameters, use the man command: `man read_ilm`
`-cell` *`cellName`*	Specifies the block name. *Data_type*: string, optional
`-oa_cellview` *`lib cell view`*	Specifies the library name, cell name, and view name in the OpenAccess database format. *Data_type*: string, optional
`-directory` *`dirName`*	Specifies the directory for the files you want to read. *Data_type*: string, optional ⚠ For OA flow, the `-directory` option is not required and `read_ilm` is used as follows: `read_ilm -cell pll_fbdiv` The flow automatically finds the OA path when the `-directory` for `read_ilm` is not set.

Examples

- The following commands create ilms and specify ilms for an OA design:

```
#create ilms for each block
write_ilm -oa_cellview {OA_PTN_LIB tdsp_core abstract}
write_ilm -oa_cellview {OA_PTN_LIB results_conv abstract}
#specify ilms in top level
read_ilm -cell tdsp_core
read_ilm -cell results_conv
```

- The following commands specifies directory `dir_dsp_core` for block `tdsp_core`, and directory `dir_onemore` for block `onemore`:

```
read_ilm -cell tdsp_core -directory dir_dsp_core
read_ilm -cell onemore -directory dir_onemore
```

- The following commands show how to reset the ILM directory from `dir_tdsp_core` to `dir2_tdsp_core`:

```
read_ilm -cell tdsp_core -directory dir_tdsp_core
read_ilm -cell tdsp_core -directory dir2_tdsp_core
```

  These commands are equivalent to the following commands, which include `reset_ilm`:

```
read_ilm -cell tdsp_core -directory dir_tdsp_core
reset_ilm -cell tdsp_core
read_ilm -cell tdsp_core -directory dir2_tdsp_core
```

Related Information

- `reset_ilm`
- `unflatten_ilm`
- `write_ilm`

report_ilm_status

```
report_ilm_status
[-help]
```

Reports ILM status information including ILM block names, ILM data locations, and if the design is in the flattened or unflattened state.

You can use this command at any time in the flow, but it only reports ILM information if you have already specified ILMs in the design.

Parameters

`-help`	Outputs a brief description of the `report_ilm_status` command. For a detailed description of the command, use the man command: `man report_ilm_status`

Example

The following command generates the ILM status report for `dtmf_recvr_core`:

```
@[DEV]ilmView 2> report_ilm_status

================================================================================
Current design: dtmf_recvr_core
Current view is ILM (after flatten_ilm)
Current ILM data is for si mode.
================================================================================

Cell                      Model-Type       Use-Type       Stage       Instance Name
--------------------------------------------------------------------------------
results_conv                 All             ILM           -          RESULTS_CONV_INST
tdsp_core                    All             ILM           -           TDSP_CORE_INST

ILM/flexILM Data Location :
--------------------------------------------------------------------------------
results_conv : .../libs/ilm/results_conv.ilm
tdsp_core : .../libs/ilm/tdsp_core.ilm
================================================================================
```

Related Information

- reset_ilm
- unflatten_ilm
- write_ilm

reset_ilm

```
reset_ilm
[-help]
-cell cellName
```

Unspecifies the ILM cell.

This is used to unspecify a previously specified ILM. This would be useful in situations where ILMs are to be removed from the design, to put them in non-ILM or black box mode.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `reset_ilm` parameter. For a detailed description of the command and all of its parameters, use the man command: `man reset_ilm`
`-cell` *`cellName`*	The name of the block to unspecify. *Data_type*: string, required

Related Information

- `reset_ilm`
- `unflatten_ilm`
- `write_ilm`

set_ilm_type

```
set_ilm_type
[-help]
[-model {timing si}]
```

Controls the behavior of commands in the presence of ILMs.

ILM models can be an SI model or a non-SI model. In the flow, sometimes an explicit `flatten_ilm` needs to be performed to perform the underlying operation. Based on the nature of operation, an appropriate model could be invoked/switched with the above command.

Use the `get_ilm_type` command to display the current settings for the `set_ilm_type` command.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `set_ilm_type` parameter. For a detailed description of the command and all of its parameters, use the man command: `man set_ilm_type`.
`-model {timing si}`	Specifies the ILM model that is to be used by the `flatten_ilm` command. All the timing-related commands require data to be in the flattened state (physical command require unflattened state). Design commands such as `opt_design` and `time_design` automatically set this parameter before flattening ILMs. However, the CTE commands and some clock commands, which can be run in the flattened mode require manual settings. • Use the `si` model for flattening the SI ILM model. • Use the `timing` model for flattening the non-SI related timing commands. *Default*: `timing` *Data_type*: enum, optional

Examples

- The following command can be used to get timing reports containing the SI push-out delays on nets using the `-set_ilm_type -model` command:

```
set_ilm_type -model si

# Flattens to SI model

flatten_ilm

# Turns on SI aware delay calculation

set_db delaycal_enable_si true

# Adds increemental delay column (for SI push-out delays)
#  in timing output:

set_global_report_timing_format {instance arc cell fanout load slew delay
incr_delay arrival}

# Minimizes the width of the report such that it easily
# fits into the screen without wrapping

set_table_style -name report_timing -no_frame -indent 0

report_timing
```

Note: You can also invoke the Global Timing Debugger (*Timing - Debug Timing - Generate*)

Related Information

- flatten_ilm
- get_ilm_type
- opt_design
- time_design

unflatten_ilm

```
unflatten_ilm
[-help]
```

Reverts the ILM back to its original hard macro view.

You can use this command after the `flatten_ilm` command.

Note: `unflatten_ilm` will take no effect if the design is already in the unflattened mode.

Many physical commands would not work in the flattened mode. They would need to be unflattened before performing the operation.

For example:

```
flatten_ilm

report_timing

unflatten_ilm
```

Parameters

`-help`	Outputs a brief description that includes the usage of the `unflatten_ilm` command.

Example

The following command flattens all partitions and their ILM designs to the top-level. After reporting timing, the design is reverted back to the unflattened state with the `unflatten_ilm` command.

```
flatten_ilm

report_timing

unflatten_ilm
```

Related Information

- `flatten_ilm`
- Top-level Timing Closure Methodologies chapter of the *Innovus Stylus Common UI User Guide*.

update_partition

```
update_partition
[-pin_location]
[-post_eco_suffix postECOSuffix]
{-flex_ilm_eco}
{-golden_block_dir directory}
-flex_ilm_dir directory
[-eco_dir directory]
[-full_netlist_with_eco_dir directory -flex_ilm_eco_dir directory]
```

Creates a new partitioned output that contains the optimized blocks and top. It also performs ECO on the original block. The ECO output is saved as new Innovus database in the partitioned block. The ECO would happen only on `flexILM` partitions and not the regular blocks.

Parameters

`-eco_dir` *directory*	Specifies the scratch directory for partitioning and running distributed ECO job after optimization. *Data_type*: string, optional
`-flex_ilm_dir` *directory*	Specifies the new partitioned output directory containing optimized blocks/top and ECO results. *Data_type*: string, optional
`-flex_ilm_eco`	Performs ECO on FlexILM modules. **Note:** This is default. *Data_type*: bool, optional
`-flex_ilm_eco_dir` *directory*	Specifies the postOpt FlexILM data directory. *Data_type*: string, optional
`-full_netlist_with_eco_dir` *directory*	Specifies the directory to store full netllist ECO result. *Data_type*: string, optional
`-golden_block_dir` *directory*	Specifies the golden database directory for individual blocks. *Data_type*: string, optional
`-pin_location`	Updates new partition pin locations into the golden block DB. This option does not allow new added logic ports. *Data_type*: bool, optional
`-post_eco_suffix` *postECOSuffix*	Specifies the suffix name that will be part of generated postECO DB name. *Data_type*: string, optional

write_ilm

```
write_ilm
[-help]
[-allow_ilm_eco]
[-ignore_ports ListOfPorts]
[-keep_all]
[-keep_selected]
[-model_type {timing si all}]
[-no_inter_clock_path]
[-overwrite]
[-remove_power_ground]
[-to_dir dirName  | -oa_cellview {lib | cell | view}]
[ [-opt_stage {prects postcts}] ]
```

Creates the specified directory containing ILM files with the appropriate level of extraction for specific model type such as timing, si, or for all models. For MMMC, ILM models will be generated based on constraint modes.

The `write_ilm` command creates a reduced set of instances representing only the interface paths, that is, paths from the I/O ports of the design to the first level of sequential instances (latches are considered sequential elements).

The command automatically modifies the module name with the following rule:

```
topModuleName+timestampl+$+moduleName
```

Parameters

`-help`	Outputs a brief description that includes type and default information for each `write_ilm` parameter. For a detailed description of the command and all of its parameters, use the man command: `man write_ilm`
`-allow_ilm_eco`	Allows ECO changes within the ILM block.
`-ignore_ports` *ListOfPorts*	Specifies list of ports where these ports and the associated interface path(s) which are starting or ending at these ports will not be included in the ILM model. *Data_type*: string, optional

`-keep_all`	Preserves the complete logic of block and hence does not do any logic pruning based on constraints or path groups. **Note:** Since the complete logic is preserved, so all the ILM views, that is, timing, CTS, and SI share the same model data. *Data_type*: bool, optional
`-keep_selected`	Preserves all selected instances and nets (in addition to those normally kept) when you create ILMs. Selected instances are those instances that drive the selected nets. You can select instances or nets using the `select_obj` command, then use `write_ilm -keep_selected`. By default, no additional instances and nets are preserved. *Data_type*: bool, optional
`-model_type {timing | si | all}`	Specifies the type of ILM model to be created. Default: `all` • `all`: Ensures that all models are written. • `timing`: Ensures that only the timing model is written for non-SI based timing, optimization, or clock tree commands. • `si`: Ensures that only the SI model is written that can be used for the SI based timing and optimization commands. *Data_type*: enum, optional
`-no_inter_clock_path`	Minimizes the clock latency at the upper-level CTS when the non-boundary flop in ILM has the maximum clock latency.
`-oa_cellview {lib | cell | view}`	Specifies OA cell view information for generated ILMs. The `-oa_cellview` option cannot be used in non-OA flow. For non-OA flow, you use the `-to_dir` option instead. *Data_type*: string, optional
`-overwrite`	Overwrites existing files. *Data_type*: bool, optional

`-opt_stage {preCTS | postCTS}`	Specifies the optimization stage, preCTS or postCTS. This option should be specified both for ILM and flexILM creation. *Data_type*: enum, optional
`-remove_power_ground`	Removes PG ports, PG nets, PG pin connections, and replaces PG net connections on signal pins with 1'b1 or 1'b0 .
`-to_dir` *dirName*	Specifies the output directory name where ILMs file will be saved. If the design is not an OA design, this option must be specified. *Data_type*: string, optional

Examples

- The following command generates both timing model and SI model in directory BlockA:
  `write_ilm -dir BlockA`
- The following command only generates timing model and saves it in directory BlockA:
  `write_ilm -dir BlockA -model_type timing`

Note: In the current ILM flow, SDC constraints originally specified against the complete flat netlist for the design get filtered when being applied against the ILM view of the design.

Note: The ilm Category Attributes also affect the instances that `write_ilm` command will preserve to model the block.

Sample Summary Report

The following is a sample summary report generated at the end of the `write_ilm` command:

```
-------------------------------------------------------------

Model                Reduced Instances         Reduced Registers

ilm_data             7153/7621 (93%)              174/285 (61%)
si_ilm_data          6793/7621 (89%)              160/285 (56%)
-------------------------------------------------------------
```

In this report, the reduction ratio in the `ilm_data` model is 93 percent which means that 7153 out the total 7621 instances for this block have been eliminated. Only 468 instances are written to the Verilog netlist for the `ilm_data` model out of which 111 instances are registers.

This summary report applies to a block using MMMC. Therefore, views with worst reduction ratio are displayed for each model.

Note: You can run the following commands for improving the reduction ratio:

```
set_db ilm_keep_high_fanout_ports false
```

Related Information

- `ilm Category Attributes`
- `read_ilm`
- "Creating ILMs for Shared Modules" in the Top-level Timing Closure Methodologies chapter of the *Innovus Stylus Common UI User Guide*.

read_ilm_eco_db

```
read_ilm_eco_db
[-help]
{-dir directoryName | -module_model_tag tagName}
```

Reads an ECO file after restoring a block-level design for non-iHDB flow.

- For iHDB flow, you should use `read_module_model -add_ons {eco}` command to read the ECO file. Internally, `reread_module_model -add_ons {eco}` will source `<cellName>_eco.tcl` script that internally calls the `read_ilm_eco_db -module_module_tag` command.
- For non-iHDB flow, if you use `write_ilm_eco_db -module_model_tag`, you should use `read_ilm_eco_db -module_model_tag` to read an ECO file at block-level design.

Parameters

`-help`	Outputs a brief description of the `read_ilm_eco_db` command. For a detailed description of the command, use the man command: `man read_ilm_eco_db`
`-dir` *directoryName*	Specifies the directory name to write the ILM eco database. At the block-level design, if you use this command to read in the ECO changes, the command will automatically find out about the design name and only load/read the ECO file for this specific design.
`-module_model_tag` *tagName*	Specifies the tag name to use for iHDB. Use this option if `write_ilm_eco_db -module_model_tag` is used to save ECO file(s).

Related Topics

- read_module_model
- write_ilm_eco_db

write_ilm_eco_db

```
write_ilm_eco_db
[-help]
-cell tcl_obj]
{-dir directoryName  | -module_model_tag tagName}
```

Saves netlist and placement information changes::

- Generates a binary ECO file that can be read/loaded back to a block-level design
- Generates the ECO TCL script with `read_ilm_eco_db -module_model_tag` for iHDB
- Output as corresponding ECO report file for viewing purpose

Parameters

`-help`	Outputs a brief description of the `write_ilm_eco_db` command. For a detailed description of the command, use the man command: `man write_ilm_eco_db`
`-cell` *`tcl_obj`*	Specifies a cell name for ECO information to be written out. If the specified cell name does not have any ECO change, both ECO and report files will be empty. If this option is not specified, the command will write ECO information for all ILMs specified in the design EVEN an ILM cell is not allowed for ECO. This to make sure accumulated ECO information will be preserved/saved as an ILM can be opened and then closed later during the flow. If there is no ECO change done to a particular cell, its ECO and report files will be empty.
`-dir` *`directoryName`*	Specifies a directory name where ECO files will be saved. All the ECO and/or report files should be saved under the convention *`dirName/cellName/cellName.eco`* and *`dirName/cellName/cellName.rpt`*, respectively.
`-module_model_tag` *`tagName`*	Specifies the model tag name where ECO file, ECO TCL, and its report will be saved. This option should be used for Stylus hierarchical DB.

Related Topics

- read_ilm_eco_db

16

Import and Export Commands and Attributes

- add_route_via_defs
- add_route_vias Category Attributes
- check_design
- check_unique
- compare_def
- create_pin_text
- create_tracks
- delete_tracks
- design Category Attributes
- disconnect_dangling_port
- Import and Export Without Category Attributes
- init_design
- init Category Attributes
- read_db
- read_def
- read_def Category Attributes
- read_mmmc
- read_name_mapping
- read_netlist
- read_physical
- read_physical Category Attributes
- read_stream
- relink_db_files
- reset_design
- run_abstract
- run_abstract Category Attributes

- set_cell_power_domain
- set_library_unit
- set_power_lib_files
- trim_design
- update_lef_macro
- write_db
- write_db Category Attributes
- write_def
- write_def_by_section
- write_def Category Attributes
- write_design
- write_do_lec
- write_lef_abstract
- write_mmmc
- write_name_mapping
- write_netlist
- write_netlist Category Attributes
- write_stream
- write_stream Category Attributes
- write_via_defs
- report_gates
- report_sequential
- check_design Category Attributes

add_route_via_defs

```
add_route_via_defs
-help
[ -ndr_only ]
```

This command is used to auto-generate vias required by nanoroute for the default and non-default rules (NDRs). The vias are based on the wire-widths of the routing-rule and are created to meet all the DRC via enclosure rules with variations for good pin-access, variations of double-cut vias for good double-cut via utilization, and min-area vias required for stacked-vias that have no route connected to meet the min-area metal rule.

The command only creates vias in the current Innovus session; they are not saved and restored for usage in a later Innovus session. So this command is normally only used for experimentation, or possibly if you just want temporary extra DFM vias to be used for this one session. If you want to save the vias for usage by nanoroute in later Innovus sessions, use `add_route_vias_auto true`. The command is not set to "on" by default as add_route_vias Category attributes are set to `false` by default.

The command requires correct DRC rules (e.g. enclosure, spacing, minimum-cut, min-area rules, etc), and at least one LEF VIARULE GENERATE statement (or OpenAccess standardViaDef) for every via layer used for routing (as required by OpenAccess). The cut enclosure values in the VIARULE GENERATE statement are not used except for old LEF technology files (LEF version 5.5 and older) that have no enclosure rules (this is not recommended usage, but is kept for backward compatability). The VIARULE cut-spacing values are ignored–only the DRC cut-spacing values are used.

No predefined vias are required (e.g LEF VIA DEFAULT, or NONDEFAULTRULE USEVIA or an OpenAccess validVias list for any routing ConstraintGroup are not needed).

Any predefined vias that do exist, are kept unless `add_route_vias_ [delete_via_before_generation` is used (see below). Generated vias are compared to predefined vias and if they are geometrically identical, that specific generated via is not kept so no duplicate vias are created.

Note: NDRs that have the same widths as the default rule will inherit the same vias, including any pre-defined vias.

Parameters

`-help`	Prints a brief description that includes the type and default information for each `add_route_via_defs` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man add_route_via_defs`.
`-ndr_only`	Specifies vias generation for non-default rules only. When specified, it generates signal routing vias for all NDR (non-default rules) and skips generating via for default rule. It is always used for designs having full set of default rule vias, but not sufficient non-default rule vias. *Data_type*: bool, optional

Example

- `add_route_via_defs`
  Will add auto-generated vias to any existing vias for all routing rules.
- The following command generates vias for all NDRs, including the NDRs created by create_route_rule command.
  ```
  add_route_via_defs -ndr_only
  ```
  You must use `add_route_vias_auto true` and `add_route_vias_ndr_only true` in the Innovus flow.

Related Information

- `write_via_defs`
- `generate_special_via Category Attributes`

add_route_vias Category Attributes

The root attributes in the `add_route_vias` category are:

- add_route_vias_auto
- add_route_vias_ndr_only

To set the value of any root attribute, use the `set_db` command:

```
set_db attribute_name value
```

To get the current value of any root attribute, use the `get_db` command:

```
get_db attribute_name
```

To get the current value of all attributes in the `add_route_vias` category, use the following command:

```
get_db -category add_route_vias
```

add_route_vias_auto

```
add_route_vias_auto {true | false}
```

Default: `false`

Data_type: bool, read/write

Enables auto via generation when design is loaded. The option is of type bool.

add_route_vias_ndr_only

```
add_route_vias_ndr_only {true | false}
```

Default: `false`

Data_type: bool, read/write

Generates vias for non default rules only.

check_design

```
check_design
[-help]
[-out_file fileName]
{[-type {library netlist power_intent timing hierarchical pin_assign budget assign_statements place
opt cts route signoff all}]
[-no_check ]}
```

Checks that all prerequisites to run all or part of the hierarchical or block implementation flow are met.

The `check_design` command checks the preconditions for major flow steps before they are run. If there are errors identified by `check_design`, the current script will stop. A complete list of errors is dumped to `-out_file`.

Some examples of problems that will cause design checks to fail include:

- Cells used in the design that cannot be legally placed
- Tracks misaligned with some or all cell pins
- In advanced processes, power nets that cannot be colored in the expected pattern once cells are placed
- Cell rows that are unblocked for placement but where pin layers are obstructed
- Cell rows that are unusable due to orientation and power rail misalignment
- Cells instantiated in the netlist that have no timing model defined
- Tracks blocked by power grid spacing constraints
- Instances not covered by a true timing path in any view
- ILM model with no clock tree defined
- No hold fixing cells are specified and delay cells are not usable
- No clock inverters, buffers, clock gates or logic cells are specified and the library cells are not usable for clock trees
- CTS cannot balance all skew groups without increasing insertion delay
- Preserved nets and cells in the clock tree that will violate other constraints because they cannot be resized or buffered.

Parameters

`-help`	Prints a brief description that includes the type and default information for each `check_design` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man check_design`.
`-type`	

-type

Specifies the category of the flow to be checked.

The intent is to run `check_design` before each of the operation in the list, such as run `check_place -place` before placement to make sure there are no issues that would cause issues during placement. Each of the categories run several checks related to that operation.

- `library`: Checks the library integrity and corrects binding of all design instances to library cells or RTL modules (hinst → module, inst → base_cell/lib_cell).
  - For physical_only library, check_design will flag libcells that have only LEF but no .lib representation.
  - For logical_only library, check_design will flag libcells that have only .lib but no LEF representation.
  - For unresolved library, check_design will check if design instance is missing binding (hinst → module, inst → base_cell/lib_cell)
- `netlist`: Checks the netlist integrity at any point in the flow.
  - For undriven netlist, check_design will check the list of input/inout hpins/pins with no driver on the net connected to it (driver can be input port or output hpin/pin).
  - For multiple_driver netlist, check_design will check the list of input/inout pins or output/inout ports with more than one driver on their net)
  - For unloaded netlist, check_design will check the input/inout ports or output hpins/pins with no load (input/inout pins) in their fanout or hinsts/insts that have nothing connected to any output pin)
  - For constant netlist, check_design will check the output ports or input pins/hpins that are driven by a logical constant 0/1.
  - For feedthrough netlist, check_design will check the hinsts, where every output pin is directly driven by an input pin, w/o any logic in between).
- `power_intent`: Checks related to MSV setup including power domain/fence checks. .
- `timing`: Performs checks related to timing setup integrity. This check is run prior to running any operation that requires timing information. **Note**: The `check_timing` command should still be used for more exhaustive timing checks.
- `hierarchical`: Performs checks related to the hierarchical flow. This type checks if a partition has a fence. If not, it will give an error. It also checks if the partition fence is outside the core boundary. If it is outside, the check will give an error. This check is run before running hierarchical design operations.
- pin_assign: Checks related to pin assignment for partitioning flows. This check is run before pin assignment.
- `budget`: Checks related to time budgeting for partitioning flows. This check is run before timing budgeting.
- `assign_statements`: Performs checks for the netlist for assign statements. This check is run

	before exporting a netlist for signoff. • `place`: Checks related to placement such as PG rail alignment, pin access issues, missing followpin in the rails of standard cells, misaligned well tap cell, and missing termination cell. This check is run before placement. • `opt`: Checks related to optimization such as checks for available cells. This check is run before optimization. • `cts`: Checks related to clock tree definition such as constraint targets and route types. This is run before cts but after creating the clock tree spec. This check cannot run with pre-cts Innovus DB (just after place_opt_design). You need to set CCOpt option (target skew, target trans, etc) and also need to load the CCOpt spec file. • `route`: Checks related to routing such as congestion, placement, and pin access issues. The `-type` route cannot be run with pre-placed DB. This check is run before routing. It is useful to run check_drc ahead of time so the design markers are up to date. • `signoff`: Checks related to the signoff activities including extraction and ECOs. This check is run before TSO. To use this types, you need to set the Tempus path, else the tool will give an error. • `all`: Checks related to all categories. **Note**: `-type` or `-no_check` is required (except when using `-help` argument) *Data_type*: (enum)+, required
`-no_check`	Just echo out all the messages that would be checked, don't do any checks Do not run the checks but echo the checks that would run per the categories provided by the user. *Data_type*: bool, required
`-out_file` *fileName*	Specifies the output report file. Contents dumped to this file are in the Tcl format. Sourcing this file will create a TCL dictionary of the violations found by `check_design`. *Data_type*: string, optional

Examples

- The following example shows the usage of the `-out_file` option:

```
> check_design -type "place cts route" -out_file checks.tcl
```

*** Summary of all messages that are not suppressed in this session:

```
Severity ID Count Summary
WARNING CHKCTS-16 1 Preferred routing layers for route_type ...
ERROR CHKCTS-1 3 CTS maximum transition target must be se...
```

```
ERROR CHKCTS-10 1 Route type(s) %s used for trunk or top c...
WARNING CHKCTS-21 1 Clock logic cells are not specified for ...
WARNING CHKCTS-46 1 Clock logic cells are not specified for ...
WARNING NRIG-1323 65 Net %s has bottom preferred routing laye...


> source checks.tcl

> dict keys $design_checks
cts route

> dict keys [dict get $design_checks place]

CHKCTS-1 CHKCTS-10 CHKCTS-16 CHKCTS-21 CHKCTS-46

# Each check as the following keys
> dict keys [dict get [dict get $design_checks cts] CHKCTS-1]

severity count violations

# For information regarding the check, use get_message:
> get_message -id CHKCTS-1 -short

CTS maximum transition target must be set to a numeric value for delay_corner %s and delay_type
%s for clock_tree %s and net_type %s.

# To query the violations, use the check dictionary:
> dict get [dict get [dict get $design_checks cts] CHKCTS-1] violations

{dc_0.9v_125c late clk top}
{dc_0.9v_125c late clk trunk}
{dc_0.9v_125c late clk leaf}


CTS maximum transition target must be set to a numeric value for delay_corner %s and delay_type
%s for clock_tree %s and net_type %s.

# To query the violations, use the check dictionary:
> dict get [dict get [dict get $design_checks cts] CHKCTS-1] violations

{dc_0.9v_125c late clk top}
{dc_0.9v_125c late clk trunk}
{dc_0.9v_125c late clk leaf}
```

- The following command generates a list of errors that will prevent or seriously impact the running of the `place` step in the implementation flow. Here is an example of the contents of the output file:

```
#############################################################
# Generated by:     Cadence Innovus 17.12-e034_1
# OS:               Linux x86_64(Host ID ip-172-19-134-167)
# Generated on:     Mon Nov  6 05:53:12 2017
# Design:           fuuBar
# Command:          check_design -type place -out_file place.checkDesign
#############################################################
```

```
#Design Check : place (12432915)
# ------------------------------------------------------------------------------
# Message: The via for net %s in DPT layer %s does not align with the tracks at %.4f %.4f.
#    Help:
# ------------------------------------------------------------------------------
dict set design_checks place IMPPP-4418 {
 count 12432915
 violations {
  {vddcx_2 M3 464.7360 2139.8625}
  {vddcx_2 M3 464.3520 2139.8625}
  {vddcx_2 M3 463.8720 2139.8625}
  {vddcx_2 M3 464.7360 2139.8825}
  ...
 }
}
#Design Check : place (7280)
# ------------------------------------------------------------------------------
# Message: The color of PG wire which is a non-default width %.4f wire for net %s in layer %s should have the opposite color with the track at %.4f
%.4f.
#    Help:
# ------------------------------------------------------------------------------
dict set design_checks place IMPPP-4420 {
 count 7280
 violations {
  {0.0800 vssx_0 M0 202.3680 1095.6000}
  {0.0800 vddcx_2 M0 202.3680 1095.3000}
  ...
 }
}
```

Related Information

- check_drc
- check_place
- check_timing

check_unique

```
check_unique
[-help]
[-verbose]
```

Checks the uniqueness of the netlist. Certain features, such as clock tree synthesis and IPO, require a module (a non-leaf cell) to be instantiated only once. If the netlist is unique, the result is `1`; otherwise, the result is `0`.

You can use the `check_unique` command after importing a design.

Parameters

`-help`	Prints a brief description that includes the type and default information for each `check_unique` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man check_unique`.
`-verbose`	Displays the names of modules instantiated more than once. *Data_type*: bool, optional

Example

The following command checks the uniqueness of the netlist and displays the names of any modules instantiate more than once:

```
innovus 4> check_unique -verbose
```

The sample output is as shown below:

```
Non-leaf cell A instantiated 2 times.
0
```

Related Information

- `check_netlist`

compare_def

```
compare_def
[-help]
in_file
[-ignore_cell_place lef_macro_classes_list [-check_cell_prefix prefix_list]]
[-ignore_layer_from layer]
[-ignore_layer_to layer]
[-ignore_logic_change]
[-place_blockage]
[-report_file in_file]
[-row]
[-special_net_pin]
[-unit units_value]
[-bump]
[-def_as_golden]
[-metal_shape_only]
```

Compares a DEF file with the physical design data in the Innovus database and reports any differences. For example, after you perform an Engineering Change Order (ECO) and make changes to a DEF file, you can use the `compare_def` command to compare the new file to the information stored in the Innovus database.

You can use the `compare_def` command after importing a design.

Parameters

`-help`	Prints a brief description that includes the type and default information for each `compare_def` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man compare_def`.

`-bump`	Checks only bump-related objects, such as bump and areaio instance in the COMPONENTS section, bump assignment, and special routing in the SPECIALNETS section.The information reported includes: • Added, moved, and deleted bumps • Added, changed, and deleted routing • Added, changed, and deleted net assignments **Note:** When this option is specified, `compare_def` suppresses checks on default objects, such as standard cells and blocks, and only checks bump-related objects. *Data_type*: bool, optional
`-check_cell_prefix` *prefix_list*	Checks instances whose instance name begins with the specified prefix. You can use this parameter only if you also specify the `-ignore_cell_place` parameter. If you specify more than one prefix, you must enclose the list in braces, for example, `{CORE PAD}`. Default: All LEF `MACRO` classes specified by the `-ignore_cell_place` parameter are ignored. *Data_type*: string, optional
`-def_as_golden`	Treats the DEF file as the golden file when comparing the DEF file to the data in the database. By default, the software compares the DEF to the data in the database and reports the differences relative to the database. For example, `ADDINST` in the report file means the instance is not in the database, but in the DEF file. When you specify `-def_as_golden`, the software reports the differences relative to the DEF file. Therefore, if the instance is not in the database, but is in the DEF file, `DELINST` is written in the report file. *Data_type*: bool, optional

in_file	Specifies the name of the DEF file you want to compare with the physical design data in the database. *Data_type*: string, required
-ignore_cell_place *lef_macro_classes_list*	Ignores the specified cell classes, for example {CORE PAD}. *Data_type*: string, optional
-ignore_layer_from *layer*	Ignores the specified metal or cut layer, and all metal and cut layers above. You can specify this parameter with the -ignore_layer_to parameter to select a layer range. You must specify metal layers as M1, M2, M3, and so on, and cut layers as V1, V2, V3, and so on. *Default:* All layers are compared. *Data_type*: (layer), optional
-ignore_layer_to *layer*	Ignores the specified metal or cut layer, and all metal and cut layers below. You can specify this parameter with the -ignore_layer_from parameter to select a layer range. You must specify metal layers as M1, M2, M3, and so on, and cut layers as V1, V2, V3, and so on. *Default:* All layers are compared. *Data_type*: (layer), optional
-ignore_logic_change	Ignores logic changes when comparing a DEF file with the physical design data in the Innovus database. *Data_type*: bool, optional
-metal_shape_only	Without the option, the command will report the same routing segment or via with one pair of line as difference - one with "ADDWIRE/ADDVIA" and one with "DELWIRE/DELVIA" - in the report file if it is on a different net or if its routing type has changed from special to regular or vice versa between db and def file. With the option, the command will not report such difference in report file. *Data_type*: bool, optional

`-place_blockage`	Compares placement blockage and block halo equivalence. *Data_type*: bool, optional
`-report_file` *in_file*	Specifies the name of the report containing the results of the comparison. *Default:* `defPhyDiff.rpt` *Data_type*: string, optional
`-row`	Compares row and site equivalence, name, location, and orientation. *Data_type*: bool, optional
`-special_net_pin`	Compares snet pins. *Data_type*: bool, optional
`-unit` *units_value*	Uses the specified `UNITS` value (typically, the `UNITS` value in the DEF file) to scale all distance values in the report file. *Default:* Uses the current database unit (DBU) value in the database, derived from the LEF information, to scale distance values. *Data_type*: string, optional

Example

- The following command compares data in the `myDEF` file with that in the database, and writes the results to `myResultsFile.rpt`:

  ```
  compare_def myDEF -report_file myResultsFile.rpt
  ```

- The following command checks rows, placement blockages, and block halos in `myDEF`. This example checks all instances except class `CORE` and `PAD`; however, it checks `CORE` and `PAD` cells whose instance name prefix is `PRE`. This example also checks all wires and vias except for those on and between layers `M2` and `M6`, and writes the comparison results to `myResultsFile.def`.

  ```
  compare_def -row -place_blockage -ignore_cell_place {CORE PAD} -check_cell_prefix PRE -
  ignore_layer_from M2 -ignore_layer_to M6 -report_file myResultsFile.def myDEF
  ```

Related Information

- Interactive ECO chapter of the *Innovus Stylus Common UI User Guide*

- Comparing Physical Design Data

create_pin_text

```
create_pin_text
[-help]
[-format {assura dracula pvs}]
pin_text_file_name
{[-nets { nets… } ]
[-cells { cells … } ]}
[-layer_map fileName ]
[-format {assura dracula pvs}]
[-layer_map fileName]
[-offset {x y}]
{{[-nets nets…] [-cells cells …]} pin_text_file_name}
```

Generates a file containing pin text information for nets and cells. You can read this file into Layout Versus Schematic (LVS) tools such as Dracula® and Assura™ for debugging problem areas.

You can use the `create_pin_text` command after you have finished placing, routing, and verifying your design in Innovus.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `create_pin_text` parameter. For a detailed description of the command and all of its parameters, use the man command: `man create_pin_text`.
`-cells` *cells …*	Specifies the list of leaf-level cells or top cell for which the pin text information is to be generated. The list of cells can be specified within braces or quotes. Additionally, pin text information can be output for all the cells using `"*"`. The `x y` values generated for the pin text are coordinates in user units local to the cell. You should not specify this parameter with the `-nets` parameter if you want to use this file with Dracula; however, you can specify both parameters if you use Assura or a third-party LVS tool. *Data_type*: string, optional
`-format {assura dracula pvs}`	Specifies the output format. *Data_type*: enum, optional

`-layer_map` *fileName*	Specifies the name of the file that contains the mapping between LEF layers and layers defined in the LVS tool. The layer map file has the following format: `LEF_LAYER_NAME VERIF_LAYER_NAME` If you do not specify this parameter, Innovus assumes that the LEF layer names and LVS layer names are identical. *Data_type*: string, optional
`-nets` *nets…*	Lists the name of the nets for which you want to generate pin text. You cannot use a wildcard (`*`) to specify all nets. You must use braces to enclose the list of nets. You should not specify this parameter with the `-cells` parameter if you want to use this file with Dracula; however, you can specify both parameters if you use Assura or a third-party LVS tool. *Data_type*: string, optional
`-offset {x y}`	Changes the design origin by -offset x y. *Data_type*: point, optional
pin_text_file_name	Specifies the pin text output file name. For information about the output file format see the Examples section below. *Data_type*: list, required

Examples

- The `create_pin_text` command creates a file containing one line of data for each cell or net you specify. The line has the following format:

  `name X= x Y= y ATTACH= verif_layer_name cell_name`

  The elements are:

name	Specifies the net name (from the `-nets` parameter) or pin name (from the `-cell` parameter).
`X=` *x* `Y=` *y*	Specifies the x and y coordinates for the pin text.
`ATTACH=` *verif_layer_name*	Specifies the layer name used in the LVS tool.

cell_name	Specifies the cell name. The default is the top cell name.

- The following command creates a file, `pins.txt`, containing pin text information for `clk1` and `VDD`:

  ```
  create_pin_text -nets {clk1 VDD} pins.txt
  ```

  The output file would resemble the following:

  ```
  clk1 X=0.0 Y=10.0 ATTACH=metal1 top_cell

  VDD X=-10.0 Y=-10.0 ATTACH=metal2 top_cell

  VDD X=20.0 Y=20.0 ATTACH=metal3 top_cell
  ```

- The following command creates a file, `hedtext.txt`, for cells `block1` and `block2`:

  ```
  create_pin_text -cells {block1 block2} hedtext.txt
  ```

  The output file would resemble the following:

  ```
  pin1b1 X=0.0 Y=0.0 ATTACH=metal1 block1
  pin2b1 X=1.0 Y=1.0 ATTACH=metal2 block1
  ```

  and so on, until all pins for `block1` are listed. Coordinates are local to `block1`.

  ```
  pin1b2 X=0.0 Y=0.0 ATTACH=metal2 block2
  pin2b2 X=1.0 Y=1.0 ATTACH=metal2 block2
  ```

  and so on, for all pins for `block2`.

- The following command creates a file, `pins.txt`, containing pin text information for `clk1`, `clk2`, and `rst` pins of the top cell `myTopCell`:

  ```
  create_pin_text -nets {clk1 clk2 rst} -cells {myTopCell} pins.txt
  ```

  The output file would resemble the following:

  ```
  clk1 X=204.27 Y=157.64 ATTACH=METAL2 myTopCell
  clk2 X=152.13 Y=0.28 ATTACH=METAL2 myTopCell
  rst X=0.33 Y=167.72 ATTACH=METAL2 myTopCell
  ```

create_tracks

```
create_tracks
[-help]
-direction direction
-layer layerNameList
-num numOfTrack
-start coordinate
-step step
```

Creates non-uniformed, customized tracks in selected design area. Typically, the routing grid is generated when the floorplan is initialized. The first track is located at an offset from the placement grid set by the OFFSET value for the layer in the LEF file. The track spacing is the PITCH value for the layer defined in LEF.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `create_tracks` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man create_tracks`.
`-direction` *direction*	Specifies track direction as a string. *Data_type*: string, required
`-start` *coordinate*	Specifies the starting coordinate to create the track. The unit for the coordinates is user unit (micron). *Data_type*: float, required
`-num` *numOfTrack*	Specifies the number of tracks to create for the grid. You cannot specify `0 numtracks`. *Data_type*: int, required
`-step` *step*	Specifies the spacing between the tracks. *Data_type*: float, required
`-layer` *layerNameList*	Specifies the routing layer used for the tracks. You can specify more than one layer. *Data_type*: string, required

Related Information

- add_tracks
- check_tracks
- delete_tracks

delete_tracks

```
delete_tracks
[-help]
[-direction {X|Y}]
[-layer layerNameList]
```

Deletes a set of tracks in the database. Without arguments, it deletes all the tracks in the database.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `delete_tracks` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man delete_tracks`.
`-direction {X|Y}`	Specifies the direction of the track to be deleted. X indicates vertical lines; Y indicates horizontal lines. *Data_type*: string, optional
`-layer layerNameList`	Specifies the routing layer on which the tracks to be deleted. You can specify more than one layer. *Data_type*: string, optional

Example

- The following command deletes all the tracks in the database:
  ```
  delete_tracks
  ```

Related Information

- `add_tracks`
- `check_tracks`
- `create_tracks`
- `report_tracks`

design Category Attributes

The root attributes in the design category are:

- design_backside_bottom_routing_layer
- design_backside_top_routing_layer
- design_bottom_routing_layer
- design_compressed_pg_db
- design_cong_effort
- design_dual_rail_via_pitch
- design_early_clock_flow
- design_express_route
- design_high_freq_interposer_flow
- design_ignore_followpin_vias
- design_merge_trim_shapes
- design_power_effort
- design_flow_effort
- design_process_node
- design_pessimistic_mode
- design_slack_weighting_method
- design_tech_node
- design_top_routing_layer
- design_trim_grid_group

To set the value of any root attribute, use the `set_db` command:

```
set_db attribute_name value
```

To get the current value of any root attribute, use the `get_db` command:

```
get_db attribute_name
```

To get the current value of all attributes in the `design` category, use the following command:

```
get_db -category design
```

design_backside_bottom_routing_layer

`design_backside_bottom_routing_layer` *layerName*

Default: ""

Data_type: string, read/write

Specifies the highest backside LEF layer name for global and detail routing.

design_backside_top_routing_layer

```
design_backside_top_routing_layer layerName
```

Default: ""

Data_type: string, read/write

Specifies the lowest backside LEF layer name for global and detail routing.

design_bottom_routing_layer

```
design_bottom_routing_layer layerName
```

Default: ""

Data_type: bool, read/write

Specifies the lowest LEF layer name for global and detail routing.

For a special partial routing on the first routing layer in certain nodes, the routing enablement of that layer must be a global decision in the entire flow of the design. For example, if M0 or M1 routing is enabled, you need to set `set_db -design_bottom_routing_layer M0/1` from the init step. This bottom routing layer setting must not be changed in later stages of the flow.

design_compressed_pg_db

```
design_compressed_pg_db {true | false}
```

Default: `false`

Data_type: bool, read/write

Specifies whether the compressed PG feature is enabled. When it is set to `true`, `read_db` will automatically compress the PG data while loading the design. You can use `get_db current_design .is_pg_compressed` to check if PG in the design is compressed.

design_cong_effort

```
design_cong_effort {low medium high auto}
```

Default: `auto`

Data_type: enum {low medium high auto}, read/write

Specifies the congestion effort level.

design_dual_rail_via_pitch

```
design_dual_rail_via_pitch min_via_pitch min_filler_via_pitch cell_boundary_spacing
```

Default: `""`

Data_type: string, read/write

This attribute is used when two horizontal power-rails on the first and third routing layer are used for standard-cells, and the min-spacing of stacked-vias between the dual-rails is controlled during placement and routing.

- *`min_via_pitch`*: It is the minimum center-to-center pitch of the stacked-vias touching any active standard-cell (not a filler cell).
- *`min_filler_via_pitch`*: It is the minimum center-to-center pitch of the stacked-vias touching only filler cells.
- *`cell_boundary_spacing`*: Specifies how close a PIN or OBS shape must be to the cell boundary to block a stacked-via in the power rail at that point.

The placer makes sure that there is room for a stacked-via as required, the global routers estimate the congestion cost before the stacked-vias are created, and the power-router adds them during the followpin routing as required.

design_early_clock_flow

```
design_early_clock_flow {true | false}
```

Default: `false`

Data_type: bool, read/write

Enables the early clock flow.

design_express_route

```
design_express_route {true | false}
```

Default: `false`

Data_type: bool, read/write

With "-flowEffort express", should routing-stage be forced.

design_high_freq_interposer_flow

```
design_high_freq_interposer_flow {true | false}
```

Default: `false`

Data_type: bool, read/write

Specifies the flow as a high frequency interposer flow. When set to `true`, `set_integration_route_constraint` will auto turn off the options which are not suitable for interposer routing.

design_ignore_followpin_vias

```
design_ignore_followpin_vias {true | false}
```

Default: `false`

Data_type: bool, read/write

Directs the detailed placer, optimization, and signal routing to ignore any special_via with the followpin shape. You can use this attribute if the two bottom routing layers both have a horizontal routing direction, and you use dual power-rails on those two layers on top of each other for the standard-cell power rails.

In this usage, other commands and options will add as many special_via as possible between the power-rails, and remove any special_via that cause spacing violations between the sViaInst cut shapes and any cell or signal routing cut shapes. This gives the placer and router more freedom to get higher density, run faster, but still put as many special_via as possible between the dual power-rails.

When set to true, followpin vias are ignored during detailed placement, optimization, and routing.

design_merge_trim_shapes

```
design_merge_trim_shapes {{layers layer_name_list gap gap_value max_merge_num number_of_max_merge ma
sks {mask_num ...}} ...}
```

Default: `""`

Data_type: string, read/write

Specifies the setting for trim shapes merge.

- `layers`: You use this t specify the trim layer list which needs to merge trim shapes.
- `Gap`: You use this setting when the end to end gap is less the value.
- `max_merge_num`: You use this to specify the max trim shapes number that are merged.
- `masks`: You use this to specify the trim shapes color that needs to be merged.

design_power_effort

```
design_power_effort {none low high}
```

Default: `none`

Data_type: enum {none low high}, read/write

Specifies the effort to reduce the dynamic and leakage power. The tradeoff between dynamic and leakage power can also be controlled.

- `none`: No power optimization is performed and power is not considered during timing optimization.
- `low`: Power is considered during timing optimization and power optimization is performed in all flow stages.
- `high`: Additional algorithms are used to minimize power consumption. Turnaround time with this option will be longer than using `low` or `none`.

design_flow_effort

```
design_flow_effort {express standard extreme}
```

Default: `standard`

Data_type: enum {express standard extreme}, read/write

Guides the effort of every super-command.

- express: Configures the flow to give the best turnaround time with good WNS/area correlation as compared to the standard flow. The flow is appropriate for prototyping.
- `standard`: Configures the flow to give the best overall combination of quality of results and full-flow turnaround time. This flow is appropriate for the majority of designs.
- `extreme`: Configures the flow to give the best quality of results at some cost in turnaround time. This flow is appropriate for designs where timing closure is challenging.

design_process_node

```
design_process_node ProcessTechnologyValue
```

Default: `90`

Data_type: int, read/write

Specifies the process technology value to set for all the applications. Units in nanometers (nm).

design_pessimistic_mode

```
design_pessimistic_mode {true false}
```

Default: `false`

Data_type: bool, read/write

Specifies that the delay calculation compatibility mode should be used for correlating to external signoff analysis tools. This mode adjusts delay calculation settings to ensure the best match to the third party engine while maintaining a degree of pessimism to help limit optimistic outliers in the external signoff analysis tool.

design_slack_weighting_method

```
design_slack_weighting_method {unity_weighting view_based_weighting}
```

Default: `unity_weighting`

Data_type: enum {unity_weighting view_based_weighting}, read/write

Specifies slack weighting to be used in the implementation flow.

design_tech_node

```
design_tech_node{N22|N12|N10|N7|N7Plus|N6|N5|N5PlusPlus|N4|N3|N3E|N2|S11|S10|S8|S7|S5|S4|S3|S2|G7|G5
|ICF|I7|I5|P1277|P1278|P1280|C12|C7|unspecified}
```

Default: `unspecified`

Data_type: enum {N22|N12|N10|N7|N7Plus|N6|N5|N5PlusPlus|N4|N3|N3E|N2|S11|S10|S8|S7|S5|S4|S3|S2|G7|G5|ICF|I7|I5|P1277|P1278|P1280|C12|C7|unspecified}, read/write

Sets the design technology node.

The node value is normally automatically set by `read_physical` while loading the LEF technology and the tool prints a message mentioning the auto-detected node. However, if the auto-detection does not work, you will see a message like this:

```
## Check design process and node:
## Both design process and tech node are not set.
```

In such cases, you will need to set the value directly using `design_process_node` and `design_tech_node`.

design_top_routing_layer

```
design_top_routing_layer layerName
```

Default: `""`

Data_type: string, read/write

Specifies the highest LEF layer name for global and detail routing.

design_trim_grid_group

```
design_trim_grid_group groupName
```

Default: `""`

Data_type: string, read/write

Specifies the group name of the set from which the trim metal grid will be used for placement and routing. If the specified group name does not exist in LEF, the tool will return an ERROR.

See the LEF documentation on the TRIMMETALTRACK keyword for more details.

disconnect_dangling_port

```
disconnect_dangling_port
[-help]
[-keep_input]
[-keep_output]
[-keep_inout]
[-partition_only]
```

Disconnects dangling nets from ports in the whole design, from the top level to the bottom level. Use `report_dangling_ports` to report the dangling ports.

You can use the `disconnect_dangling_port` command after importing a design.

Parameters

`-help`	Prints a brief description that includes the type and default information for each `disconnect_dangling_port` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man disconnect_dangling_port`.
`-keep_input`	Keeps the input dangling port connected. *Data_type*: bool, optional
`-keep_output`	Keeps the output dangling port connected. *Data_type*: bool, optional
`-keep_inout`	Keeps the inout dangling port connected. *Data_type*: bool, optional
`-partition_only`	Disconnects dangling nets in partitions only. *Default:* Disconnects all dangling nets. *Data_type*: bool, optional

Example

- The following command disconnects dangling wires in partitions only:

```
disconnect_dangling_port -partition_only
```

- The following set of commands first reports the dangling ports and then disconnects them:

```
innovus > report_dangling_ports -outfile danglingport.rpt
```

  Sample content of the file `danglingport.rpt`:

```
Port sel of module user_DFT_inst_BandA_0
Port a of module user_DFT_inst_BandA_0
Port b of module user_DFT_inst_BandA_0
Port y of module user_DFT_inst_BandA_0

innovus > disconnect_dangling_port
Port sel of module user_DFT_inst_BandA_0 is disconnected...
Port a of module user_DFT_inst_BandA_0 is disconnected...
Port b of module user_DFT_inst_BandA_0 is disconnected...
Port y of module user_DFT_inst_BandA_0 is disconnected...
4 connected dangling ports are disconnected.
```

Related Information

- report_dangling_ports

Import and Export Without Category Attributes

The root attributes in the design category are:

- ignore_max_via_stack_rule
- read_write_lef_check_uncolored_shapes
- write_def_stream_check_uncolored
- write_lec_files
- write_lec_directory_naming_style
- write_lec_dft_constraints
- write_lef_abstract_customer_header

To set the value of any root attribute, use the `set_db` command:

```
set_db attribute_name value
```

To get the current value of any root attribute, use the `get_db` command:

```
get_db attribute_name
```

ignore_max_via_stack_rule

```
ignore_max_via_stack_rule {true | false}
```

Default: `false`

Data_type: `boolean`, `read/write`

The tech lef file contains `MAXVIASTACK` syntax by default. You can use `ignore_max_via_stack_rule` to turn off `MAXVIASTACK` syntax checking. When `ignore_max_via_stack_rule` is set to `true`, the `MAXVIASTACK` rule in the tech lef file is ignored.

read_write_lef_check_uncolored_shapes

```
read_write_lef_check_uncolored_shapes {pin_only | on | off}
```

Default: `off`

Default: enum, read/write

Specifies whether to check for uncolored LEF PIN and OBS shapes on multi-mask routing layers when reading or writing LEF files. If the check is on, LEF read or write commands will error out if there are any uncolored routing layer shapes (cut layers are not checked). If set to `pin_only`, only check LEF PIN shapes and ignore LEF OBS shapes.

- During `write_lef_abstract`, if the tool finds any uncolored physical pin and if `read_write_lef_check_uncolored_shapes` is set to `on`, the tool will print an error message, and will not

write out uncolored shapes to the LEF file. However, if `read_write_lef_check_uncolored_shapes` is set to `off`, the tool will not check the uncolored object and dump all physical pins into the macro lef. In this case, the tool does not print any error message.

- When the tool reads the LEF file and finds any uncolored macro pin, and if `read_write_lef_check_uncolored_shapes` is set to `on`, the tool will print an error message and only read colored shapes into the design. However, if `read_write_lef_check_uncolored_shapes` is set to `off`, the tool will not check the uncolored object and will read the definition of all macros into the design. In this case, the tool does not print any error message.

When read_write_lef_check_uncolored_shapes is set to `auto`, the tool automatically turns on the checking feature for N5 or N3 tech node.

The `read_write_lef_check_uncolored_shapes` attribute controls the behavior of following commands or flow:

- `write_lef_abstract`
- `read_db` and init design flow
- `update_lef_macro`

`read_write_lef_check_uncolored_shapes` does not impact `write_lef_library` and read/write oa tech.

write_def_stream_check_uncolored

```
write_def_stream_check_uncolored {true 1 false 0 metal_only}
```

Default: `false`

Data_type: enum, read/write

`true` above N5 nodes and `metal_only` for N5 and below nodes

Specifies how to check and ignore uncolored shapes on DPT layers during DEF out and stream out.

- `true 1`: `write_def` and `write_stream` will go through all wires/vias/physical pins first. If the related layer has the definition for the "MASK" and "SAMEMASK" rules, tool checks the object if a color should be assigned.
- `false 0`: Ignores to check the uncolored object.
- `metal_only`: Only checks the color on the metal (including wires, physical pins, and top/bottom layer of vias).

write_lec_files

```
write_lec_files {true | false}
```

Default: `true`

Data_type: bool, read/write

Specifies whether the `write_do_lec` command should write intermediate files to the verification directory specified

by the `write_lec_directory_naming_style` attribute. If false, no files are written.

write_lec_directory_naming_style

`write_lec_directory_naming_style` *`directory`*

Default: `fv/invs/%s`

Data_type: string, read/write

Specifies the directory name where the `write_do_lec` command will write verification files when the `write_lec_files` attribute is `true`. The directory will be created if it does not already exist, and will be overwritten if the existing directory has the same name. A `%s` in the string is replaced with the design name. For example, it will overwrite the directory for the same design, but not for a different design. A `%d` in the string is replaced with a unique integer to avoid overwriting any existing directory.

write_lec_dft_constraints

`write_lec_dft_constraints` *`TCLDirectory`*

Default: `""`

Data_type: string, read/write

Specifies a TCL directory that holds the LEC DFT pin constraints from the last LEC run. It is passed forward through the `write_design` command. The `write_do_lec` command will automatically include this constraint in the specified file.

write_lef_abstract_customer_header

`write_lef_abstract_customer_header` *`header_information`*

Default: `""`

Data_type: string, read/write

Writes LEF abstract files with the customer-specified header information.

init_design

```
init_design [-help]
```

Initializes the database and ensures that the tool is ready for full execution.

You must use the following commands in the order shown below before calling `init_design`.

1. set_db to set various init root attributes as desired. Normally init_power_nets and init_ground_nets are set here. See init Category Attributes chapter for all the init root attributes. You can see the current values with: get_db init*
2. read_mmmc to read timing data from an MMMC file. This step does error checking and creates MMMC objects (analysis_views, delay_corners, etc.). It reads in any timing libraries required by the set_analysis_view command inside the MMMC file, but it defers reading SDC and other files until they are needed later. This is optional for Innovus but required for Tempus.
3. read_physical to read in physical technology and library data from LEF or OpenAccess. This is required for Innovus, and optional for Tempus.
4. read_netlist to read in the netlist from Verilog or OpenAccess. This is required, and the design and netlist objects are created at this time.
5. read_power_intent to read in CPF or IEEE 1801 files. The files are read in and various error checks are done, but no power_domain objects are created until init_design is called. This step is optional.
6. `init_design` to step through the defined MMMC objects, the netlist objects, and the power intent data to build up the full representation of the design. After initialization is complete, the design is in a consistent state and is ready for timing analysis, floorplanning, placement, etc. Any data required by active views from set_analysis_view inside the MMMC file (e.g. SDC files) is read in at this time.

In Stylus mode, `eval_legacy {init_design}` will fail with an error message because Legacy initialization is not supported in Stylus. Also, `init_design` silently sets `init_mmmc_version to 2` because only MMMC2 is supported in Stylus.

Additionally, the `init_design` command:

- Initializes the default bounding box of square size with 70% utilization
- Creates default rows, the site of row depends on the height of the cell used in the netlist
- Creates track in the box based on the tech LEF

The `init_design` command can be broken into the following sub-steps:

1. Power Domain Creation. The power_domain objects are created and then bound to the design data as defined in the CPF or IEEE1801 files. At the end of this sub-step, every instance in the system is associated with its power domain.
2. Bind Power Domains to Timing Conditions. The power_domains are bound to the correct timing_conditions (library_sets and opconds) based on the `create_delay_corner` commands in the MMMC file. In this sub-step, the active delay_corners are processed with each instance getting assigned to the appropriate

library_set and opcond.

3. Constraint Loading. The SDC files (and any other timing related files) required by the active analysis_views are loaded and error checking is done.
4. Active View Setting. The final step is to set the active views based on the `set_analysis_view` command in the MMMC file, and to perform final checks to ensure any active analysis_view is fully and completely defined. Now the design is ready for execution.

Timing Incremental Updates

After `init_design`, you can enter new MMMC commands (e.g. create_analysis_view, create_delay_corner, etc.) at the command line. These commands populate the mmmc database objects, and perform error checking to report missing data. No data load takes place during command entry. Instead, the loading of new libraries, constraints, and so on, is deferred until a set_analysis_view command that requires the new data is given.

Note: Reading new design data (netlist) and new physical data (LEF/OA) incremental updates is not allowed. After the first `init_design` is completed, these steps are effectively disabled. If you issue a `read_netlist` or `read_physical` command after the first `init_design`, the tool returns an error.

Note: The annotation data, such as SPEF or SDF, cannot be loaded if the design has not been initialized properly with the `init_design` command.

Parameters

`-help`	Prints out the command usage. For a detailed description of the command, use the `man` command: `man init_design`

Example

Here is an example of a typical init script.

```
set_db init_power_nets {VDD VDD2}

set_db init_ground_nets VSS

read_mmmc mmmc.tcl                    #Only .lib files for the active views are read at this time

read_physical -lef {tech.lef stdcell.lef}  #The technology LEF must be first

read_netlist {design.v block.v}

read_power_intent -cpf design.cpf

init_design
```

Related Information

- create_delay_corner
- init Category Attributes
- read_mmmc
- read_netlist
- read_physical
- read_power_intent
- set_db
- set_analysis_view

init Category Attributes

The root attributes in the `init` category are:

- init_check_netlist
- init_check_output_pin_constant
- init_delete_floating_hnets
- init_design_netlist_type
- init_design_uniquify
- init_ground_nets
- init_ignore_pg_pin_polarity_check
- init_keep_empty_modules
- init_lef_files
- init_min_dbu_per_micron
- init_mmmc_files
- init_no_new_assigns
- init_oa_abstract_views
- init_oa_default_rule
- init_oa_design_cell
- init_oa_design_lib
- init_oa_design_view
- init_oa_foundry_rule
- init_oa_layout_views
- init_oa_ref_libs
- init_oa_search_libs
- init_oa_special_rule
- init_oa_tech_lib
- init_power_intent_files
- init_power_nets
- init_read_netlist_allow_port_mismatch
- init_read_netlist_allow_undefined_cells
- init_read_netlist_files

- init_sync_relative_path
- init_timing_enabled

To set the value of any root attribute, use the `set_db` command:

```
set_db attribute_name value
```

To get the current value of any root attribute, use the `get_db` command:

```
get_db attribute_name
```

To get the current value of all attributes in the `init` category, use the following command:

```
get_db -category init
```

init_check_netlist

```
init_check_netlist {true | false}
```

Default: `false`

Data_type: bool, read/write

After reading in the netlist, it performs the same checks as `check_design -type netlist`.

Related Commands

- `init_design`
- `check_design`

init_check_output_pin_constant

```
init_check_output_pin_constant {true | false}
```

Default: `true`

Data_type: bool, read/write

Stops the Verilog reader if any output (or inout) signal pins have any 1'b0 or 1'b1 constants that would cause shorts to power/ground.

The default value of this boolean variable is `TRUE`.

Related Commands

- `init_design`

init_delete_floating_hnets

```
init_delete_floating_hnets {true | false}
```

Default: `false`

Data_type: bool, read/write

If `true`, an hnet (Verilog wire statement) that has nothing connected to it, is deleted from the netlist. The default value is `false` in Innovus to allow for netlist repair.

Related Commands

- `init_design`

init_design_netlist_type

```
init_design_netlist_type  netlist_source
```

Default: `Verilog`

Data_type: string, read/write

Specifies the source of the design netlist. You can specify either `Verilog` or `OA`.

Example

- The following command specifies Verilog as the source of the design netlist.
  ```
  set_db init_design_netlist_type {Verilog}
  ```

Related Commands

- `init_design`

init_design_uniquify

```
init_design_uniquify {true |false}
```

Default: `false`

Data_type: bool, read/write

Specifies whether the top is uniquified. When set to `true`, the design will be uniquified during the read and flatten process. Optimization is not allowed on a non-unique design. For use in a master/clone partitioning flow, the value should be `false`.

Example

- The following command specifies that the design should be uniquified automatically during `init_design`:

  ```
  set_db init_design_uniquify true
  ```

Related Commands

- `init_design`

init_ground_nets

```
init_ground_nets  list_of_nets
```

Default: `{}`

Data_type: string, read/write

Specifies the list of global ground nets used in the design.

Example

- The following command specifies `VSS` as the ground net used in the design.

  ```
  set_db init_ground_nets {VSS}
  ```

Related Commands

- `init_design`

init_ignore_pg_pin_polarity_check

```
init_ignore_pg_pin_polarity_check list
```

Default: `{}`

Data_type: string, read/write

Space separated list of leaf cell pin names to ignore polarity checking during `connect_global_net` and CPF operations. Allows USE POWER pin to be connected to a USE GROUND net (or vice versa).

When specified, GNC/CPF are allowed to silently make connections to PG pins that are not of the same polarity as the PG net that they are being connected to. The list of pin names is only checked for the case where a polarity check failed and then can be ignored and connected in spite of the polarity mismatch.

Example

```
set_db init_ignore_pg_pin_polarity_check "bias"
```

Related Commands

- connect_global_net

init_keep_empty_modules

```
init_keep_empty_modules {true | false}
```

Default: true

Data_type: string, read/write

By default, usage of an empty Verilog module that has no matching library cell name becomes an empty hinst. If this value is false, a dummy library cell is created, and its usage becomes an inst. You should then use other commands to 'fix' the dummy library cell before proceeding in the flow.

Related Commands

- delete_empty_hinsts
- read_db

init_lef_files

```
init_lef_files  list_of_files
```

Default: {}

Data_type: bool, read/write

Specifies the list of LEF files to be imported. This attribute cannot be used with init_oa_ref_libs.

Example

- The following command specifies technology.lef and stdcell.lef as the LEF files to be imported.

  ```
  set_db init_lef_files {technology.lef stdcell.lef}
  ```

Related Commands

- init_design
- write_db

init_min_dbu_per_micron

```
init_min_dbu_per_micron DBUPerMicron
```

Default: 0

Data_type: int, read/write

Sets the minimum database units per micron allowed when LEF files initialize the database units.

You can specify a value for the minimum database units per micron allowed when LEF files initialize the database units. This option is intended for users with LEF files that have database units that are not fine enough for NanoRoute to run properly, that is when NanoRoute requires the LEF database units to be finer than the manufacturing grid. If you do not want to modify your LEF file, you can use this option to override the LEF database units value.

The default value is `0`, which means that the software uses the LEF file units. The software uses the units set in the LEF file, if the LEF units are greater than the minimum database units you specify. Conversely, if the units you specify are greater than the LEF units, the software uses the units you specify. The value you set is saved so the next time you open the design it will have the same database units.

Related Commands

- `read_db`

init_mmmc_files

```
init_mmmc_files  file_name
```

Default: `{}`

Data_type: string, read/write

Specifies a Tcl command script that contains the analysis view configurations for multi-mode multi-corner (MMMC) timing analysis and optimization. If this view definition file is not in the current directory, specify the complete path to the file.

Example

- The following command specifies `mmmc.tcl` as the MMMC view definition file.

  ```
  set_db init_mmmc_files {mmmc.tcl}
  ```

Related Commands

- `init_design`

init_no_new_assigns

```
init_no_new_assigns {true | false}
```

Default: `false`

Data_type: bool, read/write

Turns on an assign-free flow. By default, applications like optimization can add Verilog assigns to remove unnecessary buffers and reduce power. If you have external tools that do not support Verilog assigns, you can set this value to true to prevent applications from adding any new assign statements to the Verilog netlist. If there are existing assigns in the Verilog netlist you can remove them with the `delete_assigns` command..

Related Commands

- `init_design`

init_oa_abstract_views

`init_oa_abstract_views` *`list_of_views`*

Default: `""`

Data_type: string, read/write

Specifies the OpenAccess view names that the tool should examine to find the equivalent LEF MACRO information (for example, PINS, OBS, FOREIGN). If this value is null (`""`), the tool uses `abstract` as the value.

The `read_netlist` command uses `init_oa_abstract_views` list for on-demand loading of missing cells. It is used by the `read_physical` and `read_netlist` commands.

Example

- The following command sets ABS1 as the OpenAccess view to be examined for equivalent LEF MACRO information.

  ```
  set_db init_oa_abstract_views {ABS1}
  ```

Related Commands

- `init_design`
- `read_netlist`
- `read_physical`

init_oa_default_rule

`init_oa_default_rule` *`ruleName`*

Default: `{}`

Data_type: string, read/write

Specifies a constraint group name at the start of the Innovus session. The name provided would be used by Innovus for reading the place-and-route technology rules from the constraint group.

If the user-specified constraint group is not found, Innovus outputs an error during `init_design`. If no value is specified for init_oa_default_rule, Innovus falls back to the default look-up mechanism for `LefDefaultRouteSpec` constraint group used by any place-and-route tool. Follow these guidelines while using this parameter:

- You can set this attribute only at the start of Innovus session or after running reset_design.
- You will need to resolve conflicts in a design arising due to the two incompatible constraint groups used in two design running in two different Innovus sessions.

The specified constraint group must have the following:

- Valid layers
- Valid vias
- Width
- Spacing

Related Commands

- `init_design`

init_oa_design_cell

`init_oa_design_cell` *`cellname`*

Default: `{}`

Data_type: string, read/write

Specifies the OpenAccess design cell name to be used.

Example

- The following command specifies `myOACell` as the OpenAccess database cell.

  ```
  set_db init_oa_design_cell {myOACell}
  ```

Related Commands

- `init_design`

init_oa_design_lib

`init_oa_design_lib` *`libraryname`*

Default: `{}`

Data_type: string, read/write

Specifies OpenAccess design cell name to be used. Set by the `read_netlist -oa_cell_view` command.

Example

- The following command specifies `myOALib` as the OpenAccess database library.

  ```
  set_db init_oa_design_lib {myOALib}
  ```

Related Commands

- `init_design`
- `read_netlist`

init_oa_design_view

`init_oa_design_view` *viewname*

Default: `{}`

Data_type: string, read/write

Specifies the OpenAccess design view name to be used.

Example

- The following command specifies `myOAView` as the OpenAccess database view.

  ```
  set_db init_oa_design_view {myOAView}
  ```

Related Commands

- `init_design`

init_oa_foundry_rule

`init_oa_foundry_rule` *foundry_rule_name*

Default: `{}`

Data_type: string, read/write

Specifies the foundry constraint group to be used in OpenAccess. If your technology library contains multiple `foundry_innovus_XXXX` constraint groups, use the `init_oa_foundry_rule` attribute to specify which one of them has to be read in during design initialization. This attribute is used by the init_design command.

Example

- The following command specifies `foundry_innovus_2` as the foundry constraint group to be used during design initialization:

  ```
  set_db init_oa_foundry_rule {foundry_innovus_2}
  ```

Related Commands

- `init_design`

init_oa_layout_views

`init_oa_layout_views` *`list_of_views`*

Default: `{}`

Data_type: string, read/write

Specifies the list of OpenAccess view names to be processed for cell layout viewing (GDSII equivalent).

It is also used in the case where a Verilog netlist is read and views from the `init_oa_layout_views` list are used in the Quick Abstract Inference flow to derive abstract information from layout views for cells that were not processed from the libraries in the `read_physical -oa_ref_libs` list. The `read_netlist` command uses `init_oa_layout_view` list for on-demand loading of missing cells. Used by the `read_physical` and `read_netlist` commands.

Example

- The following command specifies `LAY3` as the layout view to be imported.

  ```
  set_db init_oa_layout_views {LAY3}
  ```

Related Commands

- `init_design`
- `read_netlist`
- `read_physical`

init_oa_ref_libs

`init_oa_ref_libs` *`list_of_OA_libs`*

Default: `{}`

Data_type: string, read/write

Specifies the list of OpenAccess reference libraries to import. Each reference library is processed using the abstract view name list (`init_oa_abstract_views`). The first library in the list indicates the library that was used for

the technology information unless `read_physical -oa_tech_lib` is specified. All cells from the specified libraries are read. It is important to include the standard cell library in the list to make sure that cells that are not in the current netlist are read and available for optimization. The `init_oa_abstract_views` and `init_oa_layout_views` lists specify which views of the cells are read. Set by the `read_physical -oa_ref_libs` command.

For example, if the reference library is `"lib1 lib2"`, and the abstract view name list is `"abstract abstract2"`, LEF MACRO information is processed for `lib1` with the `abstract` view. Then, for any cells in `lib1` that do not have `abstract` but do have `abstract2`, that view is processed for MACRO information. If a cell has both views, the first one is used. The process then is repeated for `lib2`.

The `init_oa_ref_libs` attribute cannot be used `with init_lef_files`.

Example

- The following command specifies `OAREFLIB1 OAREFLIB2` as the OpenAccess reference libraries.

  ```
  set_db init_oa_ref_libs {OAREFLIB1 OAREFLIB2}
  ```

Related Commands

- `init_design`
- `init_oa_search_libs`
- `read_physical`
- `write_db`

init_oa_search_libs

```
init_oa_search_libs  list_of_OA_ref_libs
```

Default: *

Data_type: string, read/write

Specifies the list of OpenAccess reference libraries to be searched when cells are used in the Verilog netlist that are not included in the list of cells in the libraries specified in `init_oa_ref_libs` list. By default all libraries in the cds.lib are searched. The `init_oa_abstract_views` and `init_oa_layout_views` lists are used to indicate which view names are looked for during the search. Applies only when `read_netlist` command is used to process a Verilog netlist to specify the connectivity to be read. Set by the `read_physical -oa_search_libs` command and used by the `read_netlist` command.

Related Commands

- init_design
- init_oa_ref_libs
- read_netlist
- read_physical

init_oa_special_rule

```
init_oa_special_rule name
```

Default: {}

Data_type: string, read/write

Specifies the default rule to be used in OpenAccess to define the via search order for power routing. Typically used to specify a metal stack LEFSpecialRouteSpec type constraint group to read from the technology library if the library contains multiple metal stacks. If nothing is specified, then the constraint group with the name LEFSpecialRouteSpec is searched in OpenAccess. It is used by the read_physical command.

Related Commands

- init_design
- read_physical

init_oa_tech_lib

```
init_oa_tech_lib lib_name
```

Default: {}

Data_type: string, read/write

Specifies the OpenAccess library to use as a source of technology information. If not specified, the first library in the init_oa_ref_lib list will be used. Applies only when read_netlist is used to process a Verilog netlist. It is set by the read_physical -oa_tech_lib command.

Related Commands

- init_design
- read_netlist
- read_physical

init_power_intent_files

```
init_power_intent_files string
```

Default: ""

Data_type: string*, read-only

Specifies the power intent files read by the previous run of the `read_power_intent` command.

init_power_nets

```
init_power_nets  list_of_power_nets
```

Default: `{}`

Data_type: string, read/write

Specifies the list of global power nets used in the design.

Example

- The following command specifies `VDD` as the power net in the design.

  ```
  set_db init_power_nets {VDD}
  ```

Related Commands

- `init_design`

init_read_netlist_allow_port_mismatch

```
init_read_netlist_allow_port_mismatch  {true | false}
```

Default: `false`

Data_type: bool, read/write

If `false`, a Verilog instance with ports that do not match the Verilog module or library cell definition will cause an error and exit. If `true`, an error message occurs, but the extra ports are added to the module or library cell definition to allow the inconsistent netlist to be read in and debugged further. The default is `true` for Innovus to allow repair of ‘dirty netlists’, but `false` for Tempus that normally does not tolerate ‘dirty netlists’.

Related Commands

- `init_design`

init_read_netlist_allow_undefined_cells

```
init_read_netlist_allow_undefined_cells  {true | false}
```

Default: `true`

Data_type: bool, read/write

If false, a Verilog reference to an undefined cell (for example, a Verilog module name that has no matching library cell and no Verilog module definition) causes an error and an exit after parsing the netlist. If `true`, an empty Verilog module is created with a warning. The default value is `true` in Innovus to allow repair of 'dirty netlists'. An empty module will become an hinst or inst depending on the value of `init_keep_empty_modules`.

Related Commands

- `init_design`

init_read_netlist_files

```
init_read_netlist_files filenames
```

Default: `""`

Data_type: string,

A list of Verilog files read in during initialization. `read_netlist` will set this root attribute to the list of input files given to it. The original input file names are retained through the flow for reference, and used by `write_do_lec` for comparison of the default golden netlist file names.

init_sync_relative_path

```
init_sync_relative_path {true | false}
```

Default: `false` (You must change your working directory to the directory saved in the previous design session before you restore the design.)

Data_type: bool, read/write

Synchronizes all relative paths in the top.globals file to the current working directory. Use this parameter if you want to restore the design without first changing your current working directory to the directory saved in the previous design session.

Example

- The following command synchronizes all relative paths to the current working directory:
  ```
  set_db init_sync_relative_path  true
  ```

Related Commands

- read_db

init_timing_enabled

```
init_timing_enabled {true | false}
```

Default: true

Data_type: bool, read-only

If read_physical is issued before read_mmmc or read_libs in Genus and Tempus, an error will be issued. In Innovus, doing this will result in init_timing_enabled being set to false by the tool. Innovus will be in physical only mode, and any attempt to set timing data will result in an error.

read_db

```
read_db
[-help]
[-dynamic_view analysis_view]
[-hold_views hold_views]
[-leakage_view analysis_view]
[-no_binary_constraint]
[-no_timing]
[-read_globals tcl_file]
[-setup_views setup_views]
{[in_dir [-mmmc_file fileName][-lef_files list_of_files]] | [-oa_lib_cell_view {libname cellname viewname} [-mmmc_file fileName]]}
[ -no_timing_graph ]
```

Reads a saved database created by write_db in a previous session. It auto-detects the database type and loads it. An OpenAccess DB format is read with -oa_lib_cell_view {*libname cellname viewname*} while a native Innovus DB format is read using *in_dir* for the DB directory name.

The write_db command optionally includes various types of data, including timing-graph data and RC extraction data. So, the state of the DB after read_db depends on what was written, and you may need to repeat RC extraction, timing analysis, etc. if that data was not saved.

For OpenAccess usage, simple digital-only designs generally go back and forth with no special handling. But some types of Virtuoso custom data cannot be represented directly in the Innovus DB, and are stored “on the side” by read_db and geometrically equivalent data is visible in Innovus. That data is then restored by write_db when you write back to OA using the correct OA settings.

There are different types of OA methodologies and flows supported that are controlled by root attributes. Use get_db oa* to see the attributes, and look at the *Mixed Signal Interoperability Guide* for more details on their effects on write_db/read_db.

Note:

- Innovus does not support loading another DB while a design is already in memory.
- You should not modify the contents of a saved DB (*.enc.dat or *.db) at any time. If the contents of the saved DB are modified, restoring such modified DB might cause problems later or lead to DB corruption or crashes.

Parameters

`-help`	Prints a brief description that includes the type and default information for each `read_db` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man read_db`
`-dynamic_view` *analysis_view*	Specifies the analysis view for the dynamic power analysis. *Data_type*: string, optional
`-lef_files` *list_of_files*	Specifies the LEF files name for loading. Use this option if you want to override the existing LEF files in the design being restored and load different LEF files instead. • This option is intended to allow upgrading to new versions of LEF files that are compatible with the old versions. The list of all LEF files used by this design must be given. • If any pin locations or cell boundaries or OBS shapes are changed for used cells, new DRC violations may occur. • If pins are added or removed from used cells, or used cells are removed, fatal errors may occur when loading the netlist. • If the removed pins and cells are unused in the design, warnings or errors may still occur if any library-specific data was stored for those cells or pins during the design flow (such as dont_touch data) **Examples** Existing LEFs used by `write_db`: `tech.lef`, `A.lef` You can change `A.lef` to `A_v2.lef` as given below: `read_db place.dat -lef_files {tech.lef A_v2.lef}` *Data_type*: string, optional

`-leakage_view` *`analysis_view`*	Specifies the analysis view for leakage power analysis. *Data_type*: string, optional
`-oa_lib_cell_view` *`string`*	Specifies the library name, cell name, and view name for reading an OpenAccess DB design. *Data_type*: string, optional
`-mmmc_file`	Specifies the path name to a Multi-Mode/Multi-Corner (MMMC) configuration file (for example, the same file format used by `read_mmmc`) to replace the MMMC file saved by `write_db` (and all the corresponding .lib files, .sdc files, *_power_constraints.tcl etc. referenced by the MMMC file). If a new MMMC file is specified using -mmmc_file, the RC data saved by the previous session may mismatch with the new MMMC file and cause issues. If `read_db -mmmc_file` is specified, by default, the tool will not load the RC data. You need to re-extract RC after the design is loaded. **Note**: CTS configuration will not be read with `-mmmc_file`. You need to re-run `create_clock_tree_spec` after `read_db` to create a new CTS configuration against the new MMMC configuration. *Data_type*: bool, optional
`-no_binary_constraint`	Specifies not to load the binary timing constraint .tcz file even it is saved to disk in the previous session. *Data_type*: bool, optional
`-no_timing_graph`	Skips reading the timing graph even if it was saved to the disk by `write_db`. *Data_type*: bool, optional
`-hold_views` *`hold_views`*	Specifies the hold views for loading, rather than the hold views active during `write_db`. *Data_type*: string, optional

`-no_timing`	Skips reading of the MMMC file, timing libraries and timing constraints. *Data_type*: bool, optional **Note:** • Most designs that use LEF rely on the .lib information to define the bus order for MACROs that have bus bit terminals. Therefore, using the `-no_timing` option can result in the connectivity to bus ports of leaf level instances being reversed. When `.lib` information is not available, the default is for bus ports to be treated as descending (for example, [7:0]). When using Verilog modules or OpenAccess abstract views to define the bus order, the connectivity will still be correct even when `-no_timing` is specified. • Timing libraries cannot be loaded in the same session after a design is read using the `-no_timing` option. You must use `read_db` without the `-no_timing` option in a new session to load the design with timing libraries and constraints. • If you save a design after `read_db -no_timing`, it will not save timing information unless you use `write_db -add_ignored_timing`. *Data_type*:
`-read_globals` *`tcl_file`*	Specifies the UDM variables to be honored when restoring the design. You need to provide a Tcl file stating the pre-defined UDM variable names to be honored. *Data_type*: string, optional
`in_dir`	Specifies the directory from which to load the DB files. If the *`in_dir`* name ends with .tar.gz it is treated as a gzipped tar file (see `write_db -gzip` option) that is extracted into the directory first. So `read_db test.tar.gz` will create a directory named test with all the DB files inside it. *Data_type*: string, required
`-setup_views` *`setup_views`*	Specifies the setup views for loading rather than the setup views active during `write_db`. *Data_type*: string, optional

Examples

- The following command loads the saved design files from the test.dat directory:

  ```
  read_db test.dat
  ```

  For physical-only editing, you can do this to speedup read_db/write_db by ignoring the timing during read_db, and adding it back during write_db.

  ```
  read_db test1.dat -no_timing
  <physical-only editing>
  write_db test2.dat -add_ignored_timing
  ```

  Then in a new session

  ```
  read_db test2.dat       #has timing from test1.dat
  ```

  If the logical netlist was modified during editing of test1.dat, then reading test2.dat may have unpredictable timing results.

- The following command reads the `placed.dat` design but replaces the saved MMMC settings in the design with the settings in `new_mmmc.tcl`:

  ```
  read_db placed.dat -mmmc_file new_mmmc.tcl
  ```

  Then create a new CTS configuration:

  ```
  create_clock_tree_spec
  ```

- The following command reads the `"designLib top placed"` OpenAccess cellview but replaces the existing MMMC settings in the design with the settings in `new_mmmc.tcl`:

  ```
  read_db -oa_lib_cell_view "designLib top placed" -mmmc_file new_mmmc.tcl
  ```

Related Information

- `create_clock_tree_spec`
- `read_mmmc`
- `write_db`
- *Design Import and Export in Stylus* chapter in the Innovus Stylus Common UI User Guide.
- Innovus Stylus Common UI Mixed Signal (MS) Interoperability Guide for OpenAccess usage.

read_def

```
read_def
[-help]
file_name
[-add_blockage]
[-all_warning]
[-components]
[-delete_bump]
[-delete_special_route]
[-ignore_drcfill]
[-ignore_pg_connectivity]
[-nets]
[-preserve_shape]
[-remove_half_wire_extension_on_pin]
[-special_nets]
[-special_route_user_class user_class_name]
[-verilog_from_def_netlist_flow]
[-incremental_routing]
```

Loads the specified DEF file. `read_def` can be used at any step after importing a design. The typical usage is:

- Load a DEF file containing the floorplan saved by Innovus or other tool.
- Load a DEF file containing the scan chain information so that placement can do scan chain reorder.
- Load a DEF file containing the cell's placement and routing saved by Innovus or other tool.

ⓘ When you load a DEF file, you must be at the same level in the hierarchy where the DEF file was saved.

The Innovus software loads the DEF file information into the database in several ways, depending on the object:

- Tracks, gcells, rows, die area
  The software deletes all of the existing objects in the database and reads in the new information.

  Note: The `read_def` command can read a rectilinear die area, but converts it to a rectangular die area with blockages at the cut off area.
- Blockages, fills, special nets
  The software compares the physical data (such as shape and layer) to the existing data in the database. If it finds the same data in the database, it ignores the data in the DEF file. If the data does not already exist, the software adds it to the database. For special nets, the software merges wire segments in the DEF file with any existing ones in the database that overlap and have the same attributes, such as layer, width, shape, and direction. The software does not delete any existing data.
- Nondefault rules, vias
  The software adds the objects to the database if they do not already exist there. If a nondefault rules

definition with the same name already exists, the software ignores the one in the DEF file. If a via definition with the same name already exists, and the via is a fixed via, the software ignores the one in the DEF file. If a generated via definition with the same name already exists, the geometries are read, but the name might be changed. The software does not delete any existing objects of these types.

The DEF reader checks the NDR rule only if its name is "DEFAULT" and checks whether all definition of the "DEFAULT" NDR match the LEF default rule. If the NDR rule name is not DEFAULT, the tool will give the following warning message:

```
WARN: The definition of "s%" are different between the NDR Rule "DEFAULT" in DEF file and the implicit default rule in LEF File.
```

Also, DEF reader will not create a "DEFAULT" NDR in DB, and give following warning message:

```
WARN: Innovus does not allow user to specify an NDR rule named as "DEFAULT", ignore it when reading the DEF file and use LEF rule to replace it.
```

- Nets
  For each net in the DEF file, the software removes the existing regular routing (not special routing), then adds the routing from the DEF file for this net.
- Pins
  For each pin in the DEF file, the software removes the existing physical information for the pin, then adds the physical information from the DEF file for this pin. If a pin does not already exist, and it is a power or ground pin, the software adds it to the database. If the pin is not a power or ground pin, the software generates an error message.
- Groups, regions
  If a group name in the DEF file matches an existing hierarchical instance name in the database, the region's boundary constraint is attached to the hierarchical instance, overriding any existing boundary constraint. The software checks whether the group members belong to the hierarchical instance, and generates a warning message if any do not belong. If a group name does not match any hierarchical instance name, the software creates the instance group with an attached region boundary constraint, if one exists. Regions that are not used by groups in the same DEF file are ignored. The software does not remove existing groups.

  Note: You can import DEF files containing wildcards (`*`) in the `GROUPS` statement. The `read_def` command does not support multiple groups per region.
- Components
  You cannot add new logical cells. You can add new physical cells, such as:
  - Cells with `CLASS PAD`
  - Cells with `CLASS COVER` and no signal pin
  - Instance with `+ SOURCE DIST`

You can use the `read_def` command after importing a design.

Note: Do not attempt to use the `read_def -verilog_from_def_netlist_flow` command in any other Tcl scripts.

After loading the DEF netlist using the `read_def -verilog_from_def_netlist_flow` command, you can perform floorplanning, non-timing driven placement and routing, wire editing, and verification. You cannot use the DEF netlist flow for parasitic extraction, delay calculation, and timing-driven placement and routing effectively because the DEF names do not properly match the Verilog names used in timing constraints and timing analysis.

Parameters

`-help`	Prints a brief description that includes the type and default information for each `read_def` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man read_def`
`-add_blockage`	Converts undefined instances (except COVER cells) into blockages. *Data_type*: bool, optional
`-all_warning`	Lists all warning messages issued when you load the DEF file. *Default:* Limits the number of warning messages to the first 1,000. *Data_type*: bool, optional
`-components`	Reads the `COMPONENTS` section of the DEF file and ignores other sections, except when the `-nets` or `-special_nets` parameters are specified. *Data_type*: bool, optional
`-delete_bump`	Deletes all bumps in the database before reading the DEF file. *Data_type*: bool, optional
`-delete_special_route`	Deletes all special routes in the database before reading the DEF file. This includes the RDL routing as well as the power and ground special routes in the design. *Data_type*: bool, optional

`-ignore_drcfill`	Ignores the special routing with the DRCFILL shape attribute. *Data_type*: bool, optional
`-ignore_pg_connectivity`	Ignores the pg connectivity in the SPECIALNETS and NETS sections. *Data_type*: bool, optional
file_name	Specifies the name of the DEF file to load. *Data_type*: bool, optional
`-incremental_routing`	Prevents deletion of the whole routing of the signal net, and just adds the new wires and vias. *Data_type*: bool, optional
`-nets`	Reads the `NETS` section of the DEF file and ignores other sections, except when the `-components` or `-special_nets` parameters are specified. *Data_type*: bool, optional
`-preserve_shape`	Prevents creation of symbolic center-line connected routing data and preserves the routing data as-is. *Data_type*: bool, optional
`-remove_half_wire_extension_on_pin`	Truncates the half-width extension back to zero extension for all wire end points that overlap a pin. This is useful for DEF files created by `write_def -add_half_wire_extension_on_pin`, which forces all `NETS` section routing to use the default half-width wire extension. **Note:** Using this parameter can cause DRC violations. *Data_type*: bool, optional
`-special_route_user_class` *user_class_name*	
	Sets the specified user-class name on all the special routes read in from the DEF file. *Data_type*: string, optional

`-special_nets`	Reads the `SPECIALNETS` and `VIAS` sections of the DEF file and ignores other sections, except when the `-components` or `-nets` parameters are specified. *Data_type*: bool, optional
`-verilog_from_def_netlist_flow`	Reconciles the names in the DEF and Verilog files . This procedure is required if you want to retrieve any information that matches the original DEF file. *Data_type*: bool, optional

Example

- The following command loads the `TOPCHIP` DEF placement file:

  ```
  read_def TOPCHIP.def
  ```

- The following command loads the power grid from a DEF file:

  ```
  read_def -special_nets TOPCHIP.def
  ```

Related Information

- `write_def`
- `write_def Category Attributes`
- Design Import and Export Capabilities chapter in the *Innovus Stylus Common UI User Guide*
- Placing the Design chapter in the *Innovus Stylus Common UI User Guide*
  - "ScanDEF-Based Reordering Approach"
- ECO Flows chapter in the *Innovus Stylus Common UI User Guide*
  - "Pre-Mask ECO Changes from a New DEF File"

read_def Category Attributes

The root attributes in the write_def category are:

- read_def_check_mask_shifts

To set the value of any root attribute, use the `set_db` command:

```
set_db attribute_name value
```

To get the current value of any root attribute, use the `get_db` command:

```
get_db attribute_name
```

To get the current value of all attributes in the `write_def` category, use the following command:

```
get_db -category read_def
```

read_def_check_mask_shifts

```
read_def_check_mask_shifts {on | off | bypass}
```

Default: `off`

Data_type: enum {off | on | bypass}, read/write

Checks conflicts between DEF COMPONENT MASKSHIFT and MACRO MASKSHIFT values with the LEF FIXEDMASK and LAYERMASKSHIFT values, respectively, in the technology and MACRO sections. You can set the following values:

- `on`: When the variable is set as on, the the tool will check for the conflict. In case of a conflict, the tool will give an error and stop running.
- `off`: When the variable is set as off, the tool will not check for the conflict.
- `bypass`: When the variable is set as bypass, the tool will give and error message and will continue running.

Related Command

- `read_def`

read_mmmc

```
read_mmmc
[-help]
file_name
```

Reads in the Multi-Mode, Multi-Corner (MMMC) file. MMMC refers to the full set of timing libraries, constraints, and the associated modes.

When the `read_mmmc` command is run, the MMMC file is read and processed. The majority of elements in the MMMC control file is saved in the MMMC caching attributes. Except for the library sets defined by the first `set_analysis_view` command, no data is loaded into the database. This command does basic syntax checking on the MMMC commands, rejecting commands that have incorrect options and commands that are not MMMC or tcl commands. However, no consistency checking is performed. For example, if a constraint mode is used in an analysis view, but is not defined, it will not be flagged.

The MMMC file contents are cached in attributes in the database. These attributes allow you to write out a full and complete MMMC file later in the flow. In addition, the names of the MMMC files are cached for querying.

The exceptions to data set load are the libraries pointed to by the first `set_analysis_view` command in the MMMC file. These library sets are loaded into the database. This load uses the standard library parser, and includes standard library syntax checking.

The `read_mmmc` command corresponds to the *Timing* step of the Initialization flow:

In the *Timing* step, the MMMC objects are set up by populating the basic MMMC information attributes. As Synthesis needs library information before the *Design* step, the *Timing* step also loads some library information. The library data to be loaded is pointed to by the first `set_analysis_view` command in the MMMC file.

See **Example** section for more details.

At a minimum, the MMMC file must contain one each of:

```
create_library_set
create_rc_corner
create_timing_condition
create_delay_corner
create_constraint_mode
create_analysis_view
set_analysis_view
```

If the MMMC file does not contain this minimum set, an error will be issued.

Note: The library sets can be specified with absolute paths in the MMMC, or relative to the local directory or to library search path attributes. The tool first attempts to load based on the name in the MMMC. If the library is not found, the tool sweeps through the search directory list in the specified order. Only the first hit is loaded.

Parameters

`-help`	Prints a brief description that includes type and default information for each `read_mmmc` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man read_mmmc`
`file_name`	Specifies the name of the MMMC file. *Data_type*: string, required

Example

When reading an MMMC file with `read_mmmc`, the tool sets the MMMC attributes with the data specified in the file. When the first `set_analysis_view` command is processed, the tool loads in the libraries specified by this analysis view. No other data is loaded.

For example, consider an MMMC file with the following commands:

```
create_library_set -name LS1 -timing [list lib1.lib lib2.lib lib3.lib]
create_library_set -name LS2 -timing [list lib4.lib lib5.lib lib6.lib]
create_library_set -name LS3 -timing [list lib7.lib lib8.lib lib9.lib]
create_rc_corner -name QX -qrc_tech file.qrc
create_timing_condition -name TC1 -library_set LS1
create_timing_condition -name TC2 -library_set LS2
create_timing_condition -name TC3 -library_set LS3
create_delay_corner -name DC1 -timing_condition {TC1 PD2@TC2} -rc_corner QX
create_delay_corner -name DC2 -timing_condition {TC1 PD2@TC3} -rc_corner QX
create_constraint_mode -name CM1 -sdc_files cond1.sdc
create_constraint_mode -name CM2 -sdc_files cond2.sdc
create_analysis_view -name AV1 -constraint_mode CM1 -delay_corner DC1
```

```
create_analysis_view -name AV2 -constraint_mode CM2 -delay_corner DC2
set_analysis_view -setup AV1 -hold AV2
```

When the `set_analysis_view` command is processed, the tool traces backwards from that command to determine the library set. In this case, the analysis view specified by `set_analysis_view` is AV1. The AV1 analysis view points to domain corner DC1, which in turn points to two library sets, LS1 and LS2. As a result, lib1.lib and lib2.lib are loaded into the database.

In this case, lib3.lib is not loaded into the database during `read_mmmc` because it is not in the library sets required by the analysis view.

Related Information

- `create_analysis_view`
- `create_constraint_mode`
- `create_delay_corner`
- `create_library_set`
- `create_rc_corner`
- `create_timing_condition`
- `init_design`
- `read_physical`
- `read_netlist`
- `set_analysis_view`
- `write_mmmc`

read_name_mapping

```
read_name_mapping
[-help]
in_file
[-debug]
```

Reads a name mapping file which records the mapping relationship between the original HDL (RTL bit level) names in Genus to the corresponding gate-level netlist names in Genus or Innovus for primary ports or reg/wire variables, and any logical phase inversions generated by Genus.

It can be used for Power Analysis using RTL switching activities and Formal Verification with Conformal LEC to facilitate key point mapping between the RTL names and the Innovus gate-level names. Innovus will not modify the incoming string <RTL_bit_level_name> and will attach this information to the corresponding netlist pin or port objects.

The input syntax of the file is as follows:

```
add mapped point RTL_bit_level_name netlist_inst_q_pin_or_port [-invert]
```

- *RTL_bit_level_name*: It is the hierarchical name of an RTL bit-level object corresponding to primary ports or reg/wire variables using “/” for the hierarchy delimiter
- *netlist_inst_q_pin_or_ports*: It is the hierarchical name of a gate-level instance q-pin or a port
- [-invert]: It indicates a logical phase inversion has been done during optimization for this pin or port

Parameters

-help	Prints a brief description that includes the type and default information for each read_name_mapping parameter. For a detailed description of the command and all of its parameters, use the man command: man read_name_mapping
-debug	Print debug info. *Data_type*: bool, optional
in_file	Name of input name mapping file. *Data_type*: string, required

Example

The following example show how different bits are merged into a multi-bit cell, so using simpler names is better.

```
add  mapped  point  IL_reg[1]      CDN_MBIT_reg/Q1      -invert
```

```
add  mapped  point  TRNS1_reg[0]   CDN_MBIT_reg/Q2
add  mapped  point  Vflag_reg      CDN_MBIT_reg/Q3
add  mapped  point  Zflag_reg      CDN_MBIT_reg/Q4
```

Example of the name mapping format used by Genus in presence of complex typed-names:

```
add mapped point clk clk //in
add mapped point sel sel //in
add mapped point in1[1] in1[1] //in
add mapped point in1[0] in1[0] //in
add mapped point out1[1] out1[1] //out
add mapped point out1[0] out1[0] //out
add mapped point out3/x[1] out3[x][1] //out
add mapped point out3/x[0] out3[x][0] //out
add mapped point isub/out1/x[1] isub_out1_reg[x][1]/Q
add mapped point isub/out1/x[0] isub_out1_reg[x][0]/Q
add mapped point out1[1] out1_reg[1]/Q
add mapped point out1[0] out1_reg[0]/Q
add mapped point out3/x[1] out3_reg[x][1]/Q
add mapped point out3/x[0] out3_reg[x][0]/Q
```

Related Information

- write_name_mapping

read_netlist

```
read_netlist
[-help]
{{in_file | -oa_cell_view {lib cell view}} [-top top_module_name]} | {-def def_file_name}}
```

Reads in a Verilog structural netlist. A structural Verilog file contains only structural Verilog-1995 constructs, such as module and gate instances, concurrent assignment statements, references to nets, bit-selects, part-selects, concatenations, and the unary ~ operator.

`read_netlist` supports reading an encrypted Verilog file. The file can be created by command `encrypt -verilog`. To be loaded, the encrypted Verilog must have suffix ".vp", like mydesign.vp. `read_netlist` also supports OpenAccess through the `-oa_cell_view` option.

The `read_netlist` command corresponds to the *Design* step of the Initialization flow:

During the *Design* step, the design is loaded into the database with `read_netlist`. After issuing this command, the database objects are populated and can be queried and manipulated (ungrouped, renamed, and so on). Full error checking is performed on the design details. This includes, but is not limited to, unresolved references, port mismatches, and empty modules.

Parameters

`-help`	Prints a brief description that includes the type and default information for each read_netlist parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man read_netlist`.
`-def` *`def_file_name`*	Specifies a DEF file to read for the netlist. This option cannot be used with any Verilog files or OA cell views. The DEF netlist is a "flat" netlist with no Verilog logical hierarchy. You can perform floorplanning, non-timing driven placement, routing, wire editing, and DRC checking. You cannot use the DEF netlist for parasitic extraction, delay calculation, and timing analysis because the flat DEF names do not match the hierarchical Verilog names used in timing constraints and timing analysis. *Data_type*: string, required
`in_file`	Specifies the list of netlist files. *Data_type*: string, required
`-oa_cell_view {`*`lib cell view`*`}`	Specifies the name of the design object to be loaded in {lib cell view} format. No other options can be used if this option is specified. *Data_type*: string, optional
`-top` *`top_module_name`*	Specified the top module name. By default, `read_netlist` identifies the highest module and links to that. If there are multiple top modules, the command returns an error. *Data_type*: string, optional

Example

- The following command reads the netlist with `dtmf_recvr_core` as the top module:

```
read_netlist mydesign.v -top_module dtmf_recvr_core
```

Related Information

- encrypt
- init_design
- read_mmmc
- read_physical
- write_netlist
- write_netlist Category Attributes

read_physical

```
read_physical
[-help]
{[-oa_ref_libs lib1 [lib2] ... [-oa_search_libs lib1 [lib2] ...][-oa_tech_lib lib]] |[-
add_oa_ref_libs lib1 [lib2]... [-oa_lib_path libPath]] | -lefs lib1 [lib2] ... |  -add_lefs lib1
[lib2] ...}
```

Reads the physical library information. When this command is run, the physical library information is loaded into the database. Both LEF and OA are supported, but not simultaneously.

The `read_physical` command corresponds to the *Physical* step of the Initialization flow:

During the *Physical* step, the physical libraries (OA or LEF) are loaded into the tool's database using `read_phsyical`. When the command is issued, the database is loaded with the specified objects, and is available for querying of attributes associated with the physical data, such as size, layer details, and so on.

The tool uses the standard OA parser, and performs full error checking on the physical data. When the command is complete, a full timing library set (via `read_mmmc`) and a physical library set would have been loaded into the database.

If `-lef` is used, the technology LEF must be first specified.

This command is optional for Synthesis and STA, but is required for Implementation.

The tool can support a physical only mode when the attribute `init_physical_only` is set to `true`. This attribute must be set prior to `read_physical`, otherwise an error will be issued.

Parameter

`-add_lefs` *lib1 [lib2]*	Specifies a list of LEF files to read and to add to the original list of LEF files that were added before `init_design`. This option can only be used after `init_design`, and cannot be combined with any other option. You can add new macros, non-default rules, sites etc. using this option, but you cannot use it to modify any existing LEF data that is already loaded. The LEF files added using this option will be used when the saved design is loaded again using `read_db`. *Data_type*: string, required
`-add_oa_ref_libs` *lib1* [*lib2*]	Specifies the list of OpenAccess library names to be loaded incrementally. These library names are added in the `cds.lib` file, if not present. Use this parameter to specify standard cell libraries or I/O driver libraries that should all be loaded after the design is read. The specified library names are appended to the `init_oa_ref_libs` root attribute. *Data_type*: string, required
`-help`	Prints a brief description that includes type and default information for each `read_physical` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man read_physical`
`-lefs` *lib1* [*lib2*] ...	Specifies the list of LEF files to be loaded. The first LEF file must have the technology section present. The file names are saved to the `init_lef_files` root attribute. This option cannot be combined with `-oa_ref_libs`. **Note**: This option can only be used before `init_design` or `read-db`. If you specify this option after `init_design`, the tool will give an error. *Data_type*: string, required

`-oa_ref_libs` *lib1* [*lib2*] ...	Specifies the list of OpenAccess library names to be loaded. These names must also be listed in the `cds.lib` file. All the abstract views in the specified libraries are loaded. Use this parameter to specify standard cell libraries or I/O driver libraries that should all be loaded, even if some of the cells are not currently used in the netlist. The first OpenAccess library that is specified is also used to load the OpenAccess technology data. The specified library names are saved to the `init_oa_ref_libs` root attribute. *Data_type*: string, required
`-oa_lib_path` *libPath*	Specifies the path of oa reference libraries. This option can only be used with the `-add_oa_ref_libs` option. *Data_type*: string, required
`-oa_search_libs` *lib1* [*lib2*] ...	Specifies the list of OpenAccess libraries to be searched. This parameter can be specified only with `-oa_ref_libs`. As the netlist is read, these libraries are searched for cell abstracts to be load on demand for any Verilog module that does not have a lib_cell already defined. This parameter is typically used for libraries of block IP, such as RAMs, where only the abstracts that are actually used in the netlist need to be loaded. The specified library names are saved to the `init_oa_search_libs` root attribute. *Data_type*: string, optional

`-oa_tech_lib` *lib*	Specifies the OpenAccess library name to be loaded. This name must also be listed in the `cds.lib` file. This parameter can be specified only with `-oa_ref_libs`. Use this parameter to specify the library that is used to load the OpenAccess technology data. If not specified, the first library specified in the `-oa_ref_libs` list will be used. The specified library names are saved to the `init_oa_tech_lib` root attribute. *Data_type*: string, optional

Examples

- Use the following command before `init_design` to load LEFs:

  ```
  read_physical -lefs {tech.lef macro_a.lef macro_b.lef}
  ```

- Use the following command after `init_design` to add LEFs to the original LEF list:

  ```
  read_physical -add_lefs {macro_c.lef macro_d.lef}
  ```

- Use the following commands to load an OA cellview and `gpdk045_innovus` is the technological library,

  ```
  set_db init_power_nets  {dvdd avddlf avdd avddhf avdd_h VDDsw}
  set_db init_ground_nets {dgnd agndlf agnd agndhf agnd_h}
  read_mmmc scripts/LP_pll.viewDefinition.tcl
  read_physical -oa_ref_libs {gpdk045_innovus gsclib045}
  read_netlist -oa_cell_view {zambezi45_top LP_pll layout}
  ```

- Use the following commands to load a netlist

  ```
  set_db init_power_nets  {dvdd avddlf avdd avddhf avdd_h VDDsw}
  set_db init_ground_nets {dgnd agndlf agnd agndhf agnd_h}
  read_mmmc scripts/LP_pll.viewDefinition.tcl
  read_physical -oa_ref_libs {gsclib045} -oa_tech_lib {gpdk045_innovus} -oa_search_libs
  {zambezi45}
  read_netlist digital_block.v
  ```

Related Information

- `init_design`
- `read_mmmc`
- init Category Attributes

read_physical Category Attributes

The root attribute in the `read_physical` category is:

- read_physical_check_antenna
- read_physical_check_colored_shape
- read_physical_check_mask_shifts
- read_physical_extend_cell_obs_shapes_under_trim
- read_physical_merge_multi_port_to_single_port
- read_physical_must_join_all_ports
- read_physical_extend_polygon_cell_shapes

To set the value of any root attribute, use the `set_db` command:

```
set_db attribute_name value
```

To get the current value of any root attribute, use the `get_db` command:

```
get_db attribute_name
```

To get the current value of all attributes in the `read_physical` category, use the following command:

```
get_db -category read_physical
```

read_physical_check_antenna

```
read_physical_check_antenna {true | false}
```

Default: `true`

Data_type: boolean, read/write

If you set `read_physical_check_antenna` to `true`, LEF or OA cells will have process antenna consistency checks done for LEF ANTENNA* data (or OA equivalent data) attached to their pins. Input pins with 0 gate-area, output pins with 0 diff-area, and inout pins with 0 gate-area or 0 diff-area values will generate warnings. If `false`, the tool will not perform consistency checking.

read_physical_check_colored_shape

```
read_physical_check_colored_shape {true | false}
```

Default: `false`

Data_type: boolean, read/write

Specifies to check and ignore colored shapes on colorless layers while reading the LEF or DEF data. When set to `true`, the tool errors out if a colored shape is on a colorless layer. When set to `false`, the tool gives error message and ignores the color.

read_physical_check_mask_shifts

```
read_physical_check_mask_shifts {on | off | bypass}
```

Default: `off`

Data_type: enum {on | off | bypass}, read/write

Checks conflicts between the LEF FIXEDMASK and LAYERMASKSHIFT values in the technology and MACRO sections when reading in any LEF files. You can set the following values:

- `on`: When the variable is set as on, the the tool will check for the conflict. In case of a conflict, the tool will give an error and stop running.
- `off`: When the variable is set to off, the tool will not check for the conflict.
- `bypass`: When the variable is set as bypass, the tool will give and error message and will continue running.

read_physical_extend_cell_obs_shapes_under_trim

```
read_physical_extend_cell_obs_shapes_under_trim listOfRoutingLayers
```

Default: `""`

Data_type: string

Specifies a list of routing layers. On each specified *layer* the LEF OBS shapes that are overlapped partially by a shape on the TRIMMETAL layer (that trims the routing layer), will be extended in the preferred routing direction to be fully covered by the trim shape. You can use this attribute for some advanced node libraries to avoid trim/OBS short violations.

read_physical_merge_multi_port_to_single_port

```
read_physical_merge_multi_port_to_single_port {true | false}
```

Default: `false`

Data_type: bool

When set to `true`, multiple ports in a LEF PIN are merged into a single port. However, the pin should not have the `MUSTJOINALLPORTS` attribute as it specifies that all of the ports must be connected individually.

read_physical_must_join_all_ports

```
read_physical_must_join_all_ports {true | false}
```

Default: `true`

Data_type: bool, read/write

If set to `false`, the LEF_MUSTJOINALLPORTS property on a LEF PIN auto-creates MUSTJOIN pins for each

PORT. If set to `true`, the MUSTJOIN pins are not created, but the router must connect to every port with a signal wire or signal via that physically overlaps all the ports.

read_physical_extend_polygon_cell_shapes

```
read_physical_extend_polygon_cell_shapes { {layer extension_amount}… }
```

Default: `""`

Data_type: string, read/write

Specifies a list of layer and a value for creating obs shape for 2D pin/obs shapes. Cells could have 2D objects, which require larger line-end spacing. You can use `read_physical_extend_polygon_cell_shapes` to automatically add SPACING 0 OBS for routing during LEF in.

For any 2D shapes, you can extend edges that are perpendicular to the preferred routing direction of the rectilinear shape by specifying the extension_amount in `read_physical_extend_polygon_cell_shapes`.

Related Command

`read_physical`

read_stream

```
read_stream
[-help]
-layer_map layer_map_files
{[-pvs_fill ] | [[in_files] [-cells cell_list [-offset {x y}]]]}
```

Reads in-stream files or a layer_map and outputs a summary of the cell names read in, and any layer/data types that have data with no map_file entry. The data is automatically scaled during reading to the correct DB units based on the stream units.

- It reads in the metal fill stream data (with top cell name and offset) to overlay on the design. The fill data is displayed to database fill shapes (equivalent to reading DEF FILL data) to allow editing or changing the fill data.
  Reading in fill data is done once per session.
- It reads in-stream cells that have the same names as Innovus cell names to display the stream cell data (for example, overlay or toggle between stream shapes and LEF shapes for the cells).
  If a stream cell name is repeated, it replaces any previous data for that cell.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `read_stream` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man read_stream`.
`-cells` *`cell_list`*	Reads only the specified cells from GDS files. *Data_type*: string, optional
`in_files`	Lists GDS and OASIS files. *Data_type*: list, optional
`-layer_map` *`layer_map_files`*	Specifies the layer map file map GDS layer to design layer. *Data_type*: string, required
`-offset` *`x y`*	Specifies, in microns, the offset for the top-level fill cell. *Data_type*: point, optional
`-pvs_fill`	Specifies to process the PVS FILL data in the read process. *Data_type*: bool, optional

Example

Parse stream cells with specified layer map display stream cell data on top of LEF cell data. The report output is redirected to read_stream.rpt.

```
>read_stream {stdcells.gds memory.gds} -layer_map gds_layer.map > read_stream.rpt
```

Read in metal-fill data and overlay it on the design. The report will be in the log file.

```
>read_stream fill.gds -cell top_fill -layer_map gds_layer.map
```

Example of output report

```
#Cell names read in:

top_fill

fill_1

fill_2

…

#Layer/dtype data in the stream file with no map-file entry

1    10

4    1, 9
```

...

Layer Map File Format

The `-layer_map` file maps stream layer/dtype data to Innovus layer and objects. Stream data does not have net ids, so you can only map stream data for display, or to fill, fill_opc or fill_blockage shapes.

The format for the layer map file is one line for each stream layer and dtype:

layer data_type layer_name object_type

Where

- *layer* is an stream layer number. It is an integer.
- *data_type* is a stream data_type number. It must be either an integer, or a range (e.g. 0-255).
- *layer_name* is normally must be a LEF (or OA tech) layer name. It can only be a "custom" layer name if the obj_type is "display" (e.g. this data is not on any of the LEF layers, but is useful for display purposes). Any custom layer data will be displayed like data created with add_gui_shape or add_gui_text. The *layer_name* is case-sensitive.
- *object_type* is one of these keywords:
  - `fill`: Equivalent to DEF FILL shapes
  - `fill_opc`: Equivalent to DEF FILL with + OPC shapes
  - `fill_blockage`: Equivalent to DEF BLOCKAGE + FILLS shapes
  - `display`: Is only for display purposes, and cannot be flattened.

  `fill`, `fill_opc`, and `display` can optionally be followed by
  `:mask:`*`mask_num`*
  where `mask_num` is the mask number for the shape for a multi-mask layer (equivalent of `DEF + MASK` *`mask_num`* for a shape).
  fill_blockage can optionally be followed by
  `:name:`*`blockage_name`*
  where *`blockage_name`* is the name given to all the blockages from this layer/dtype.

  Any text on the same layer/data_type will be mapped to the same DB layer as a text object on that layer. About the TEXT font and size, font 0 could be default (or other font decided by RD). Size could use the original size in GDS and scale it if dbu mismatch to Innovus.
  The *object_type* keywords are case insensitive. The tool ignores blank lines or lines beginning with #.
  A given layer/dtype combination can only appear once.

Example 1:

```
#Stream map-file for metal fill, via-fill and fill blockages

#only metal1 has two masks
```

```
#layer    dtype     layer_name     obj_type
1         1         metal1         fill:mask:1
1         2         metal1         fill:mask:2
51        1         cut1           fill
2         1         metal2         fill
52        1         cut2           fill
3         1         metal3         fill
…
101       1         metal1         fill_blockage:name:blkg1
101       2         metal1         fill_blockage:name:blkg2
101       3-255     metal1         fill_blockage
```

Example 2:

```
#Stream map-file for cell-data to display on top of LEF cell data
#Only routing layers and the cell-outline on custom layer “outline”
#are mapped. It does not show cut layers or layers below metal1.
#layer    dtype     layer_name     obj_type
1         0-255     metal1         display
2         0-255     metal2         display
...
#custom layer “outline” is created if it does not already exist
100       1         outline        display
```

Related Information

- write_stream
- write_stream Category Attributes

relink_db_files

```
relink_db_files
[-help]
-db_dir db_directory
-new_lib_dirs {new_lib_directory1 new_lib_directory2 ...}
```

Relinks symbolic links to library files inside a DB directory.

By default, `write_db` saves the location of library files as symbolic links under `<db_dir>/libs/…`. When a DB directory is copied to a new network where all the libraries are in a different location, or a new version of the library files are needed, you can use `relink_db_files` to change the symbolic links to new locations .

The `relink_db_files` command will search each `-new_lib_dirs` directory and the sub-directory tree below it for each library file name inside `<db_dir>`, and create a symbolic link to the new location. Only the base file name is used for the search (except for `qrcTechFile`). Therefore, two path names inside the new directory, such as `/timing/fast/std.lib` and `/timing/slow/std.lib`, will have duplicate names of "`std.lib`". The duplicate names are not supported and will cause an error. You can use the `read_db` command to update all the LEF full-path names or use a new MMMC files with all the new .lib full-path names if duplicate file names need to be updated. If there are missing library file names, the tool will show an error.

The `relink_db_files` command supports `.lib`, `.lef`, and `qrcTechFile` library file types.

Parameters

`-help`	Prints a brief description that includes the type and default information for each `relink_db_files` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man relink_db_files`
`-db_dir` *db_directory*	The current db directory which contains incorrect library symbolic links.
`-new_lib_dirs` *{new_lib_directory1 new_lib_directory2 ...}*	The new directories to search for library files.

Example

If the current LEF files are all under`/libs/lefs_v1.0/` (or sub-directories) and the current `.lib` and `qrcTechFiles` are all under`/libs/timing_v1.0/` (or sub-directories), you use the following command to update the LEF files to use `/libs/lefs_v2.0/`:

```
>relink_db_files -db_dir my_db -new_lib_dirs {/libs/lefs_v2.0 /libs/timing_v1.0}
```

Note: The command must find every library file. Therefore, you need to list the top of the old timing directory tree

along with the top of the new LEF files directory tree.

Related Information

- write_db
- read_db

reset_design

```
reset_design [-help]
```

Removes library data and design data from the Innovus database. This allows you to reload the design in the same session while debugging some problem and provides faster debugging when repeatedly loading files to debug problems inside one session.

It does not reset the internal state of applications or global settings, so it is not reliable enough to use in a real flow. For example, it does not free collections but only invalidates them. If you try to access collections saved in TCL variables, you will not be able to see those collections. If you have saved pointers to DB objects in Tcl variables, they will be invalid and may even cause a crash if used.

If you have any unexpected problems, you should start a new session without using `reset_design`.

Parameters

`-help`	Prints a brief description of the `reset_design` command usage. For a detailed description of the command and all of its parameters, use the `man` command: `man reset_design`

Example

The intended usage model is given below:

```
init_design (or read_db)

<read some files that have problems (such as a Tcl script, MMMC file, LEF file)>

<fix the files based on error messages>

reset_design

init_design (or read_db)

<read modified files to see if they are ok now>
```

If there are still errors, make more modifications, and go back to `reset_design`.
If the final result is good, you should start over in a new session and repeat it without `reset_design` to continue with the flow.

Related Information

- `init_design`
- `read_db`

run_abstract

```
 run_abstract
[-help]
-stream_files file_list
[-generate_script_only file.il]
[-lib_files file_list]
[-output_lef_file file.lef]
[-run_script_only file.il]
[-verilog_stub_files file_list]
```

Runs abstract to generate LEF from GDSII. The sequence of flow for the `run_abstract` command is as follows:

1. Checks if the executable “abstract” exists in the path. If not, it generates an error.
2. Generates “appropriate” configuration file.
3. Executes abstract with the configuration file.
4. Checks that the “abstract” exits with an exit code of “0”. If not, it generates an error.

The `run_abstract` command is run before loading a design.

The options for the `run_abstract` command are set using the run_abstract Category Attributes.

Note:

1. This command does not fully supports the block LEF abstract. Run 'ABSTRACT' standalone to generate LEF from block gds.
2. This command is under beta control. Contact your Cadence representative for more information related to this command.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `run_abstract` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man run_abstract`
`-stream_files` *file_list*	Lists the GDSII files that need to be translated **Note**: It is important that only GDSII files of the same type are listed together. For example, std-cells, IO cells, and block cells should be listed in separate runs. *Data_type*: string, required
`-generate_script_only` *file.il*	Generates the SKILL script that can be given to the abstract tool but does not execute it. *Data_type*: string, optional
`-lib_files` *file_list*	Lists liberty (.lib) files. The .lib files can also be used to define pin directions by using the direction from the matching .lib cell and pin names. *Data_type*: string, optional
`-output_lef_file` *file.lef*	Outputs the LEF file that will have the LEF macro abstracts but not technology file information. *Data_type*: string, optional
`-run_script_only` *file.il*	Does not re-generate the script but just executes it with the specified file. *Data_type*: string, optional
`-verilog_stub_files` *file_list*	Defines the pin directions by using the direction from the matching Verilog module and pin names. *Data_type*: string, optional

Related Information

- run_abstract Category Attributes

run_abstract Category Attributes

The root attributes in the run_abstract category are:

- run_abstract_abstract_blockage_cut_around_pin
- run_abstract_antenna_connectivity
- run_abstract_antenna_diffusion_geometry
- run_abstract_antenna_gate_geometry
- run_abstract_blockage_detailed_layers
- run_abstract_boundary_layers
- run_abstract_cell_symmetry
- run_abstract_export_lef_version
- run_abstract_extract_layers_power
- run_abstract_extract_layers_signal
- run_abstract_extract_pin_layers_power
- run_abstract_extract_pin_layers_signal
- run_abstract_input_cell_type
- run_abstract_input_stream_layer_map_file
- run_abstract_input_lef_tech_file
- run_abstract_keep_output_files
- run_abstract_pins_analog_names
- run_abstract_pins_clock_names
- run_abstract_pins_ground_names
- run_abstract_pins_output_names
- run_abstract_pins_power_names
- run_abstract_pins_text_pin_map
- run_abstract_pre_abstract_script
- run_abstract_selected_cells
- run_abstract_site_name
- run_abstract_verbose

To set the value of any root attribute, use the `set_db` command:

```
set_db attribute_name value
```

To get the current value of any root attribute, use the `get_db` command:

```
get_db attribute_name
```

To get the current value of all attributes in the `run_abstract` category, use the following command:

```
get_db -category run_abstract
```

run_abstract_abstract_blockage_cut_around_pin

`run_abstract_abstract_blockage_cut_around_pin` *layer_list*

Default: ""

Data_type: string, read/write

Specifies that obstruction should be cut around pin for standard cells.

By default, `run_abstract` creates obstructions on pin shapes.

- If `run_abstract_input_cell_type` is `std`, the default value for `run_abstract_abstract_blockage_cut_around_pin` is the list of all metal and via layer names. This means that abstract will not cover obstruction on pins for standard cells on all metal and via layers if `-abstract_blockage_cut_around_pin` is specified.
- If `run_abstract_input_cell_type` is `block` or `io`, the default value for `run_abstract_abstract_blockage_cut_around_pin` is the list of all via layer names. This means that abstract will not cover obstruction on pins for blocks and io pads on all via layers if `-run_abstract_abstract_blockage_cut_around_pin` is specified.

run_abstract_antenna_connectivity

`run_abstract_antenna_connectivity {`*layer1 connect_layer layer2*`}`

Default: ""

Data_type: string, read/write

Specifies the layers for which connectivity is to be extracted. The syntax is `{`*layer1 connect_layer layer2*`}`.

run_abstract_antenna_diffusion_geometry

`run_abstract_antenna_diffusion_geometry {{`*layer_name*`} {`*logical_expression*`}}`

Default: ""

Data_type: string, read/write

Specifies the diffusion layer, followed by the expression to remove the gate channel area from the diffusion region.

run_abstract_antenna_gate_geometry

```
run_abstract_antenna_gate_geometry
```

Default: ""

Data_type: string, read/write

Specifies the poly layer that forms a gate along with the layer expression to derive the gate area. This option also triggers the creation of antenna models.

run_abstract_blockage_detailed_layers

```
run_abstract_blockage_detailed_layers layer_list
```

Default: ""

Data_type: string, read/write

Specifies the layers for which a detailed blockage model is to be created. The detailed blockage model generates blockages only at those places on the layer where there are real shapes in the cell. This only applies to the IO and block cells and not to standard cells.

run_abstract_boundary_layers

```
run_abstract_boundary_layers layer_list
```

Default: ""

Data_type: string, read/write

Specifies the drawn layers to extract the cell boundary. If not specified, abstract uses the prBounbdary object as the boundary size.

By default, *layer_list* list is empty, and the prBoundary is used. The prBoundary comes from an explicit GDS layer that is used to compute the LEF SIZE (and OVERLAP OBS shapes for rectilinear boundaries). If there is no explicit GDS prBoundary layer, the boundary can be computed from the bounding box of geometries on this list of layers.

Note: This option will not normally work for core standard cells because they almost always have diffusion and metal shapes extending outside the real prBoundary. These shapes are usually shared with the row above or below for the power rails. This option can be useful for blocks that have no prBoundary and where a little lost area due to worst-case bounding box approximations is acceptable.

run_abstract_cell_symmetry

```
run_abstract_cell_symmetry {R0 | X | Y | R90 | X Y | X R90 | Y R90 | X Y R90}
```

Default: {X Y}

Data_type: string, read/write

Specifies a property value for cell symmetry. The valid values for the property symmetry are R0, X, Y, R90, X Y, X R90, Y R90, and X Y R9.

run_abstract_export_lef_version

```
run_abstract_export_lef_version {5.3 | 5.4 | 5.5 | 5.6 | 5.7 | 5.8}
```

Default: `5.7`

Data_type: enum {5.3 5.4 5.5 5.6 5.7 5.8}, read/write

Specifies the version of LEF to be exported. The default version is the same as the version of the technology LEF file being imported.

run_abstract_extract_layers_power

```
run_abstract_extract_layers_power layer_list
```

Default: `""`

Data_type: string, read/write

Specifies the layers through which the extractor extracts each power net. This is used to reduce the search space if abstract is taking too much run time for each cell.

Usually not applicable to standard cells. For blocks, it is common to only extract the power nets on the top one or two metal layers.

run_abstract_extract_layers_signal

```
run_abstract_extract_layers_signal layer_list
```

Default: `""`

Data_type: string, read/write

Specifies the layers through which the extractor extracts each net. This is used to reduce the search space if abstract is taking too much run time for each cell. Usually not applicable to standard cells.

run_abstract_extract_pin_layers_power

```
run_abstract_extract_pin_layers_power layer_list
```

Default: `""`

Data_type: string, read/write

Determines the shapes, which are found by the extractor, to be turned into pins. This argument is valid only if the associated layer, geometry specification and connectivity have been provided in the technology LEF. Usually not applicable to standard cells.

run_abstract_extract_pin_layers_signal

```
run_abstract_extract_pin_layers_signal layer_list
```

Default: ""

Data_type: string, read/write

Determines the shapes, which are found by the extractor, to be turned into pins. This argument is valid only if the associated layer, geometry specification and connectivity have been provided in the technology LEF. Usually not applicable to standard cells.

run_abstract_input_cell_type

```
run_abstract_input_cell_type {std | io | block}
```

Default: std

Data_type: enum {std io block}, read/write

Selects the cell type being translated with abstract. The tool adjusts its settings on the basis of the selected cell type.

run_abstract_input_stream_layer_map_file

```
run_abstract_input_stream_layer_map_file file.map
```

Default: ""

Data_type: string, read/write

Specifies the file that maps GDS layer number/data type to LEF layer names in the Virtuoso format.

For example, the GDS file is not colorized, you could use the common Virtuoso format GDS layer map to stream the GDS file.

```
#layer
name #purpose name #stream layer #stream data type
V0 drawing 35 0
V0 drawing 35 235
V0 drawing 35 236
M1 drawing 15 0
M1 drawing 15 235
M1 drawing 15 236
M1 drawing 15 239
```

```
For example, the gds file is colorized, you could specify the
mask color in gds layer map to stream gds file.
#layer name #purpose name #stream layer #stream data type #photo mask color #color state
V0 drawing 35 0
V0 drawing 35 235 mask1Color unlocked
V0 drawing 35 236 mask2Color unlocked
```

```
M1
drawing 15 0
M1 drawing 15 235 mask1Color unlocked
M1 drawing 15 236 mask2Color unlocked
M1 drawing 15 239 mask3Color unlocked
}
```

run_abstract_input_lef_tech_file

`run_abstract_input_lef_tech_file` *`file.lef`*

Default: ""

Data_type: string, read/write

Specifies the technology LEF file. This is used to get all the layer names and their types, such as routing, cut and the layer order for extracting connectivity.

run_abstract_keep_output_files

`run_abstract_keep_output_files {true , false}`

Default: false

Data_type: bool, read/write

Specifies to keep the temporary files/libraries that are created in the process after the `run_abstract` command finishes.

run_abstract_pins_analog_names

`run_abstract_pins_analog_names` *`pattern`*

Default: ""

Data_type: string, read/write

Identifies analog pin names based on label strings in the abstract to set the "USE ANALOG" property of LEF.

run_abstract_pins_clock_names

`run_abstract_pins_clock_names` *`pattern`*

Default: ""

Data_type: string, read/write

Identifies clock pin names based on label strings in the abstract to set the “USE CLOCK” property of LEF.

run_abstract_pins_ground_names

`run_abstract_pins_ground_names` *pattern*

Default: ""

Data_type: string, read/write

Identifies ground pins based on label strings in the abstract to set the “USE GROUND” property of LEF.

run_abstract_pins_output_names

`run_abstract_pins_output_names` *pattern*

Default: ""

Data_type: string, read/write

Identifies output pins based on label strings in the GDS to set the direction to OUTPUT property of LEF.

run_abstract_pins_power_names

`run_abstract_pins_power_names` *pattern*

Default: ""

Data_type: string, read/write

Identifies power pin names based on label strings in the abstract to set the “USE POWER” property of LEF.

run_abstract_pins_text_pin_map

`run_abstract_pins_text_pin_map` *pattern*

Default: ""

Data_type: string, read/write

Maps text labels to pins on nets. Specify the layer purpose pairs based on which you want the abstract generator to search for geometry of any given layer-purpose pair with a text label.

run_abstract_pre_abstract_script

`run_abstract_pre_abstract_script` *file.il*

Default: ""

Data_type: string, read/write

Specifies a skill file that will be executed in abstract before the auto-generated script is executed. Optional.

run_abstract_selected_cells

`run_abstract_selected_cells` *`pattern`*

Default: `""`

Data_type: string, read/write

Selects cells to create abstract for using regex style expressions.

run_abstract_site_name

`run_abstract_site_name` *`site_names`*

Default: ""

Data_type: string, read/write

Specifies the site to be used for macros. This site is added to all the cells.

run_abstract_verbose

`run_abstract_verbose {true , false}`

Default: `false`

Data_type: bool, read/write

Specifies the verbose mode with extra print.

set_cell_power_domain

```
set_cell_power_domain
[-help]
[-reset]
[-file file_name]
```

Defines domains for multi-VDD and multi-VSS cells. When reading an instance ASCII power file to perform power-rail analysis, if there are missing power values associated with ground pins, the software uses the information specified with the `set_cell_power_domain` command to derive domain association and distributes power to each ground net accordingly.

This domain information is stored in a side file that is used by power analysis to calculate power using the correct voltage for each domain. If you are specifying an external instance ASCII power file with pin information, this domain information is used to correctly distribute power to each pin.

Use this command before performing power and rail analysis.

Parameters

`-file file_name`	

	Specifies the name of the file containing the cell names and corresponding power-ground pairs.
	The format of the file is:
	`CELL: <cell name1>`
	`<power1> <gnd1>`
	`<power2> <gnd2>`
	`CELL: <cell name2>`
	`<power1> <gnd1>`
	`...`
	Data_type: string, optional

`-help`	Outputs a brief description for the `set_cell_power_domain` parameter. For a detailed description of the command, use the `man` command: `man set_cell_power_domain`.
`-reset`	Resets all specified options back to default values. *Data_type*: bool, optional

Examples

- The following command define two domains, `VDD:VSS` at `1v` and `VDDm:VSSm` at `0.84v`, of the cell `mycell` in the file `domain_file`:

  ```
  set_cell_power_domain -file domain_file
  ```

  The following is a snippet from `domain_file`:

  ```
  CELL: mycell
  VDD_1v  VSS_1v
  VDDm_0.84v  VSSm_0.84v
  ```

Related Information

- power Category Attributes
- `report_power`
- "Static Power, IRdrop and EM Analysis" in the *Voltus Stylus Common UI User Guide*
- "Dynamic Power and IRDrop Analysis" in the *Voltus Stylus Common UI User Guide*

set_library_unit

```
set_library_unit
[-help]
[-cap capUnit]
[-time timeUnit]
```

Sets the time and capacitance unit values for constraints and report generation.

The timing constraints (SDC) files do not contain any unit specification for capacitance and time values. Hence, the units for these values are considered to be same as those specified in the timing libraries. The commands, e.g., `report_timing`, require consistent units to generate reports. If an inconsistency occurs in the timing libraries, the default unit of time (as ns) and capacitance (as pf) is used.

You can use this command when you need to override the library units, or force the setting of units when libraries have different units.

Note: You must set the `set_library_unit` command before the design and constraints are read.

Parameters

`-help`	Prints a brief description that includes the type and default information for each `set_library_unit` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man set_library_unit`.
`-cap` *capUnit*	Specifies the capacitance load unit in the timing library. *Default:* `1.0` pf *Data_type*: string, optional
`-time` *timeUnit*	Specifies the time unit to be used for timing constraints. The valid values are 1ns, 1ps, 10ps, 100ps. *Default:* `1.0` ns *Data_type*: string, optional

Example

- The following command sets the time unit for the design to `10`ps:

  ```
  set_library_unit -time 10ps
  ```

Related Information

- `report_timing`
- Timing Analysis chapter in the *Innovus Stylus Common UI User Guide*

set_power_lib_files

```
set_power_lib_files
[-help] [<in_file <file1[file2] ...>>] [-reset]
```

Specify the name of the timing library file(s) for dynamic power analysis. This command allows you to run the dynamic power analysis flow without reading a dotlib into design database to reduce background memory consumption. You must have an external TWF file to run this flow as the timing engine will not be invoked from within the software. This flow is recommended for large designs where front-end processing of timing libraries is high. Some of the database query commands will not work if `set_power_lib_files` is used.

This command works in conjunction with the `set_power_def_files` and `set_power_spef_files` commands.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `set_power_lib_files` parameter.
in_file <file1[file2] ...	
	List of timing library files. *Data_type*: string, optional
`-reset`	Resets all specified options back to default values. *Data_type*: bool, optional

Examples

- The following Tcl shows the use model of the **set_power_lib_files** command that works in conjunction with the set_power_def_files/set_power_spef_files flow to perform dynamic power analysis:

```
read_libs -lef tech1.lef tech2.lef
set_power_lib_files /../library1.lib /../library2.lib
set_power_def_files adder_routed.def
set_power_spef_files {test.spef }
read_twf test.twf2
read_netlist ../design/postRouteOpt.enc.dat/super_filter.v.gz -top super_filter
init_design
set_default_switching_activity -combinational_clockgate_ratio 1 -period 2.01ns -input_activity
0.2 -integrated_clock_gate_ratio 1
set_db power_default_supply_voltage 1.1 power_method dynamic_vectorless
report_power
```

trim_design

```
trim_design
[-help]
[-core_to_boundary distance]
[-dir output_dir_to_save_trimmed_design]
[-encrypt_name proc_name_to_encrypt]
[-name trim_design_name]
[-nets list_of_net_names]
[-place_only]
[-area {x1 y1 x2 y2} | -polygon list_of_points | -selected_route_blockage ]
```

Allows you to specify the coordinates of a box to create a small portion of the design as an independent design. This trimmed down version of the design can be then used for investigation and analysis. When you share the trimmed design for investigation of a localized issue, you do not need to share the whole third-party IP information, thereby ensuring confidentiality of the third-party data.

While the trimmed design completely preserves the physical data, it does not preserve the timing and the power domain data. Working on a small portion of the design offers a faster turnaround time and a lesser memory footprint.

Notes:

- The `trim_design` command retains the attributes and properties of all the standard cells and macros that are completely inside the trim box/area. All hard macros that overlap with the trim box/area are also preserved. It does not include any standard cells or macros that are completely outside the specified trim box/area.
- The `trim_design` command preserves all overlapping rows, sites, and colors.
- The `trim_design` command flattens all instances and net names. All instances in the trim box/areal lose their original logic hierarchy and appear ‘flat’ in the trimmed database. Any naming conflicts arising due to flattening are automatically resolved. For example, h1/h2/i1 in the full design will appear as i1 in the trimmed design. Also, h1/h4/i1 may appear as i1_uniquify1, to distinguish it from i1.
  Note: To resolve naming conflicts due to the loss of hierarchy, additional name uniquification is done. A mapping of original nets/instances names vs their trimmed design modified names, is written in a map file for book keeping purpose.
- The `trim_design` command maintains the design parameters such as instance location/orientation, regular/special wire and its connectivity relative to the original full design.
- The `trim_design` command retains the attributes and properties of all the nets having some instances inside the trim box/area. It also preserves (pushes down) the nets that are not connected to any instance in the trim box/area, but have some wires overlapping with trim box/area. Nets having all instances and wires outside of trim box/area are discarded.
- The `trim_design` command creates ports for all wires crossing the boundary of the trimmed design.
- The `trim_design` command preserves the tracks, attributes and colors of all wires that are completely inside the trim box/area. All wires that overlap with the trim box/area are also preserved. It does not include any wire that is completely outside the specified trim box/area.

- The `trim_design` command preserves the Power/Ground/Special wires that overlap with the boundary of the trim box by cutting them against the boundary. It creates additional PG ports at cross points these wires.
- The `trim_design` command discards fences, regions, guides, instance groups, and power domains.
- The `trim_design` command saves the `.globals` files. The `.globals` file has the .lef list copied from original full design's .globals file.
- The `check_place` command reports all violations that were there on instances in the box in the actual design. However, since hierarchical fences are not brought back, the trimmed design may not have the Region/Fence and Not-of-Fence Violations.
- The `check_drc` command reports all violations that were there on nets in the box of actual design. However, since only the net wires crossing box boundary are preserved, the trimmed design may have some extra violations.

Parameters

`-help`	Prints a brief description that includes type and default information for each trim_design parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man trim_design`
`-area {x1 y1 x2 y2}`	List of coordinates to represent a box for getting the trimmed design. **Note**: The specified box is snapped to the instance grid. All instances in the box lose their original logic hierarchy and appear flat in the new db. *Data_type*: string, required
`-core_to_boundary` *distance*	Specifies that any instance lying within the distance of coreToBoundary (in microns) to the trim box will be discarded. *Default*: 2 *Data_type*: string, optional
`-dir` *output_dir_to_save_trimmed_design*	Specifies the name of the directory in which the data related to the trimmed-design is saved. *Data_type*: string, optional
`-encrypt_name` *proc_name_to_encrypt*	Specifies that the names of instances and nets in the design will be encrypted when the trimmed design is saved. *Data_type*: string, optional

`-name` *trim_design_name*	Specifies the top-level name of the new trimmed design. *Data_type*: string, optional
`-nets` *list_of_net_names*	Species the nets that are to be completely enclosed by the specified shape by taking union of shape with net-boxes. *Data_type*: string, optional
`-place_only`	Specifies that only instances will be pushed. *Data_type*: bool, optional
`-polygon` *list_of_points*	List of coordinates to represent a polygon for getting the trimmed design. *Data_type*: string, required
`-selected_route_blockage`	Specifies that the selected routing blockages will be used to infer the box or polygon for getting the trimmed design. *Data_type*: bool, required

Examples

- The following command creates a trimmed design, `trim_db`, with the specified polygon coordinates and save the data in the DB file.

  ```
  trim_design -dir DB -name trim_db -polygon {{500.0 700.0} {500.0 900.0} {500.0 900.0} {700.0 900.0} {700.0 900.0} {700.0 800.0} {700.0 800.0} {800.0 800.0} {800.0 800.0} {800.0 700.0} {800.0 700.0} {500.0 700.0} }
  ```

- The following command creates a trimmed design, based on the selected routing blockage:
  ```
  trim_design -dir DB -name trim_db -selected_route_blockage
  ```

- The following command uses a `get_new_name` procedure to encrypt the names on the instances and nets in the trimmed design `trim_db`:
  ```
  trim_design -dir DB -name trim_db -area {300.0 200.0 800.0 700.0} -encrypt_name gen_new_name
  ```

  The following is a structure of the encryption procedure file. You can customize this procedure:

  ```
  proc gen_new_name {name} {
    global n
    if {![info exists n]} {
      set n 0
    }
    set new_name en_name__$n
    incr n
    return $new_name
  }
  ```

Related Information

- `check_place`
- `check_drc`
- Design Import and Export in Stylus chapter in the *Innovus Stylus Common UI User Guide*
  - Trimming the Design

update_lef_macro

```
update_lef_macro
[-macros {listOfMacros}]
lef_file
```

Replaces the current MACRO information in the design with the MACRO information in the specified LEF file. You can use the update_lef_macro command to change the geometry information of a macro, such as boundary size, pin location, pin layer, and pin size. You cannot use the command to change the number or name of pins for a macro.

The update_lef_macro command only reads and replaces the MACRO information from the LEF file; it does not use any other information in the file.

Parameters

lef_file	Specifies the LEF file to read. *Data_type*: string, required
-macros {*listOfMacros*}	Replaces the information for the specified macros only. *Default:* Replaces the information for all macros in the LEF file. *Data_type*: string, optional

Related Information

- LEF Syntax chapter in the *LEF/DEF Language Reference*

write_db

```
write_db
[-help]
[-add_ignored_timing]
[-current_top_cell_only]
[-skip_file {metric cmd}]
[-no_fill]
[-no_pvs_fill]
[-no_wait out_file]
[-timing_graph [-no_constraint ]]
[{{dir_name [[-def][-verilog][-common]]} |
{{-oa_lib_cell_view {libname cellname viewname} | -oa_view {OAViewName}}[-oa_used_libs_only | -
oa_libs ]}}
[[-sdc ] [-rc ]]
[-gzip ]
[-lib]
[-lib_to_ldb]
[[-user_path [-keep_input_path ]] ]
```

Writes the complete design database in the native Innovus DB format if `<out_dir>` is given, or as an OpenAccess database (OA DB) if `-oa_lib_cell_view` or `-oa_view` is given.
The native Innovus DB format in `<out_dir>` must be used when you have LEF files for technology and physical library data in your original initialization script (e.g. `read_physical -lefs <files>`), while the OA DB output must be used if you use OA libraries (for example, `read_physical -oa_ref_libs <libs> -oa_search_libs <libs>`) during initialization.

There are different types of OA methodologies and flows supported that are controlled by root attributes. Use `get_db oa*` to see the attributes, and look at the *Mixed Signal Interoperability Guide* for more details on their effects on `write_db`/`read_db`.

You can write a design to the same location from which you read the design. The new data will overwrite the previous data.

You can use the `write_db` command at any time after `init_design`.

By default, external file references used inside *<out_dir>* (e.g. to library files, or SDC files that are not modified) are stored using a portable format that allows you to copy *<out_dir>* anywhere in the same network where the full path names are still valid, and then read it with `read_db`. This portable format is not supported for OA DBs that require the usage of a cds.lib file to find physical file locations.

Note:

- The `write_db` command does not save the state of the `timing_enable_simultaneous_setup_hold_mode` timing attribute. When the attribute is set to `true`, various physical design functions in the system are disabled. When the design is restored (using the `read_db` command), the physical design commands are required to be active so that the system can be properly initialized.
- You should not modify the contents of a saved DB (*.enc.dat or *.db) at any time. If the contents of the saved

DB are modified, restoring such modified DB might cause problems later or lead to DB corruption or crashes.

Parameters

`-help`	Prints a brief description that includes the type and default information for each `write_db` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man write_db`.

`-add_ignored_timing`	If you use `read_db -no_timing` to read the DB, the timing data is not loaded. This option allows you to copy the ignored timing data from the original `read_db` location and add it back during `write_db` like this:
	
	`read_db -no_timing test1`
	
	`<physical editing commands ...>`
	
	`write_db -add_ignored_timing test2`
	
	This usage model is intended to allow faster physical-only editing, like DRC fixing, where the netlist is not modified. If you change the netlist, unpredictable results may occur when reading the `test2` DB using timing data copied from `test1`.
	
	Data_type: bool, optional

`-current_top_cell_only`	Specifies to only write current design module and its submodules in the netlist. By default, all the modules from the input netlist will be saved even if they do not belong to current design. *Data_type*: bool, optional
`-oa_lib_cell_view {`*`libname cellname viewname`*`}`	Saves the design in OpenAccess database format using the specified lib, cell, and view names. The design is saved in the *lib/cell/view* OA directory. **Note**: Usually, the cell name is the same as the design top cell name. If the cell name specified is different from the top cell name, it means that the top module created in OpenAccess would have the new cell name instead of the original name. *Data_type*: string, optional

`-common`	Saves the common-db design data to the specified directory. **Note**: Tool does not save the common-db when LEF is not available. *Data_type*: bool, optional
`-def`	Saves a DEF file, in addition to the normal DB files in the DB directory. This parameter is normally not used for flat or block flows and is not used by `read_db`. It can be useful in a hierarchical flow, where you want to use `assemble_design` to read only the netlist and the DEF files without opening up each design DB and writing out the Verilog and DEF files separately (see `-verilog`). *Data_type*: bool, optional
dir_name	Specifies the name of the output directory to write the DB using native Innovus formats. It is common to add a `.dat` extension to the directory name. This cannot be used with OA-based designs. *Data_type*: string, required
`-keep_input_path`	Saves the path same as the path in the input file. *Data_type*: bool, optional

`-lib_to_ldb`	Converts the `.lib` timing libraries referenced in the MMMC file to the `.ldb` format and writes them in a libs sub-directory inside the output design directory. References to the original .lib files in the MMMC file are changed to the corresponding `.ldb` files inside the libs sub-directory. This makes the design directory much larger because it has all the timing library data inside it and is not recommended for normal flows. Normally a user should setup `.ldb` files in an external library directory and have the MMMC files use them directly to avoid this overhead. Using `.ldb` files will make `read_db` faster when it reads the library data, so this option is useful for doing “throw-away” debugging experiments that require repeated sessions at the cost of a one-time conversion, and carrying the library data with the design. See the `write_ldb` command for more information about the usage of `.ldb` files. *Data_type*: bool, optional

`-lib`	Normally external file references, such as to the LEF, .lib, SDC, files are saved using full path-name, symbolic links to the original files inside a libs sub-directory inside the DB directory.This option forces copies of the files rather than symbolic links in the libs sub-directory so there are no external file references. That makes the DB directory much larger, but it is now suitable for sending to a different network where the library and external data files are not available. An OA DB will still have OA library references using the normal cds.lib methodology, but any other external file references will be copied inside the cellview directory. **Note:** Any future sessions that use `read_db` and then `write_db` will have references to files inside this design directory rather than to the original external locations. So, you must keep this DB directory around as long as any other DB references it (or you must use `write_db -lib` in the later session to force another local copy of the external files). *Data_type*: bool, optional
`-no_constraint`	This option requires the -timing_graph option, and saves the timing graph data without the SDC constraints data. This is useful to reduce the disk size, and speedup write_db, read_db. But the timing-graph data cannot be loaded by any other release version, and without the SDC files, the timing constraints would be lost if loaded by a different release version. Note, you can still load the timing-graph with the current release version, and then use `write_db` without the `-no_constraint` option, to regenerate the SDC files from the timing-graph and then use read_db with a later release. *Data_type*: bool, optional

`-no_fill`	Does not write any metal or via fill shapes inside the DB. This is useful if you did ECO routing, and the fill data is going to be regenerated anyway, so it is not worth saving the current fill data any more. All the fill data that would go to DEF FILLS section (floating fill data), or the DEF SPECIALNETS section with + SHAPE FILLWIRE or FILLWIREOPC will be ignored and not written. *Data_type*: bool, optional
`-no_pvs_fill`	Does not save the PVS fill data file. In some PVS flows, a stream (GDS) file from PVS is saved inside the Innovus DB directory (see the PVS documentation for more details). During read_db this file is not read, but can be optionally loaded or used later in the flow. By default, if the PVS fill file exists, write_db will copy it from the read_db location to the new DB directory. If ECO changes to the current design have made the PVS fill file invalid, use this option to stop it from being copied. *Data_type*: bool, optional

`-no_wait` *`out_file`*	Writes the design in a separate process. You can continue executing new commands simultaneously while the design is being saved. This is useful when you want to save intermediate results without waiting for the save to complete. Simultaneous commands that modify the DB will have no effect on the saved design. The memory is shared unless you modify it, so there is little or no increase in the memory usage for small changes, but if you completely changed everything in the DB, you would temporarily have up to double the total memory for both processes to run. Each process may use multiple CPUs up to the limits set by `set_multi_cpu_usage -local_cpu`. Therefore, if the simultaneous commands use multiple local CPUs, you may temporarily have up to double the max local CPU usage until `write_db` is complete. The actual number of CPUs used by `write_db` varies with design size and the type of data being saved. So, you would need to experiment with your specific design to find the number of CPUs used, or set the `-local_cpu` value temporarily lower to limit it. The output messages or error messages which normally go to stdout/stderr (e.g. the xterm) generated by `write_db` will be redirected to *out_file*. Any .log or .logv information is ignored and lost. You can check *`out_file`* to see if `write_db` completed successfully. If you give the `exit` command, the session will wait for `write_db -no_wait` to complete before exiting, while other commands will run simultaneously. This allows flow scripts that run multiple sessions to know any save from the previous session is complete before the next session uses it, without the need to poll or check *out_file*. **Example:** The following command writes the design into the `my_design.dat` directory, and any messages are written to `save.1`. `write_db my_design -no_wait save.1` *Data_type*: string, optional

`-oa_libs`	Saves all OA libraries specified in cds.lib under the directory libname/cellname/viewname/libs/oa/. **Note**: This parameter works only in OA designs. *Data_type*: bool, optional
`-oa_used_libs_only`	Saves only the libraries listed in the `init_oa_tech_lib`, `init_oa_ref_libs`, and `init_oa_search_libs`, which are referenced through the on-demand processing. On specifying this parameter, a new cds.lib file is created with only the specified libraries. The libraries are saved under directory libname/cellname/viewname/libs/oa/. **Note**: This parameter works only in OA designs. *Data_type*: bool, optional
`-rc_extract`	Writes RC extraction data, including data for incremental extraction. By default, Innovus does not automatically save RC extraction data to save disk space and run-time. The -rc_extract parameter does not save RC extraction data if the design is not extracted or if extract_rc_engine = pre_route. *Data_type*: bool, optional
`-sdc`	By default, if the SDC timing constraints have not been modified, a link to the original external SDC file is kept to reduce the disk size. If the SDC timing constraints were modified, then the SDC files are automatically stored in the DB directory. This option will force all SDC files to be copied locally inside the DB directory, even if they did not change. **Note:** Any future sessions that use read_db and then write_db will have references to SDC files inside this design directory rather than to the original external location. So, you must keep this DB directory around as long as any other DB references it (or you must use write_db -sdc in the later session to force another local copy of the SDC files). *Data_type*: bool, optional

`-skip_file {metric cmd}`	Removes the specified db files. • If you save a design with the `-skip_file {metric}` option, the tool does not save file db `inn.um.gz` and .metric.gz. • If you save a design with the `-skip_file {cmd}` option, the tool does not save the `inn.cmd.gz` file. • If you save a design with the `-skip_file {metric cmd}` option, all `inn.um.gz`, `.metric.gz`, and `inn.cmd.gz` files cannot be saved.
`-timing_graph`	Saves the timing graph data. When this parameter is specified, the `-rc_extract` parameter is also automatically enabled. This means that the RC extraction data is also saved along with the timing graph data at the post_route stage. At the pre_route stage, the RCs are not saved but all information is present to fully time the design without the need to rebuild the timing graph. Commands that use timing analysis results like `report_timing`, `opt_design`, etc. all use the saved timing graph. This can save significant CPU after `read_db`. A user can get the same timing seen previously without waiting for the timing analysis to be repeated. **Note**: This feature is dependent on the build version and tool. Tempus cannot read a timing-graph written by Innovus. You must write and read the timing graph using the same tool and build. You can still load the design with a different tool or build, but the timing-graph is lost and you must repeat the timing analysis. The timing graph data must be fully built (e.g. with `report_timing)` before calling `write_db -timing_graph`. *Data_type*: bool, optional

`-gzip`	Saves the design as a gzipped tar file. write_db writes *<out_dir>* and automatically turns on -libs and -sdc to force all external file references to be copied inside the DB directory (e.g. .lib, .lef, .sdc files, etc). Then it tars the directory and gzips it into a single portable Innovus design tar ball, by adding the .tar.gz extension to the *<out_dir>* name. So, `write_db myDesign.dat -gzip` results in a file named myDesign.dat.tar.gz. *Data_type*: bool, optional

`-oa_view` *OAViewName*	Specifies the name of the OA view. This view will be saved in the same lib/cell/ directory that you used with `read_db` or with `read_netlist` during initialization. This option provides a shortcut to save a view quickly in the same lib/cell structure that you read from.
	Example:
	`read_db -oa_lib_cell_view {designLib top layout}`
	`write_db -oa_lib_cell_view {designLib top place}`
	`write_db -oa_view place`
	Both the above `write_db` commands are equivalent.
	Data_type: string, optional

`-user_path`	By default all external file references (such as. library files) are stored as absolute paths without any symbolic-links. So, the DB directory is portable anywhere on the same network. This option stores external file references exactly as you entered it For example, full paths like /x/y or relative paths like ../x/y, ./x/y), and keeps any symlinks used inside the path names.
	Data_type: bool, optional
	Example:
	Consider that there are two library files a.lib and b.lib under:
	`/global/project1/libs/mmmc`
	And two lef files c.lef and d.lef under:
	`/global/project1/libs/lef`

There are two symlinks to the libs:

```
/usa/db1/libs1.0/mmmc/a.lib
→ /global/project1/libs/mmmc/a.lib

/usa/libs/mmmc/b.lib           →
/global/project1/libs/mmmc/b.lib

And two symlinks to the lefs:

/usa/db1/libs1.0/lef/c.lef
    → /global/project1/libs/lef/c.lef

/usa/libs/lef/d.lef            → /global/proj
ect1/libs/lef/d.lef
```

Your current work dir is /usa/db1.1/. You have an init script like:

```
read_mmmc ./mmmc.tcl

read_physical -lefs
{/usa/db1/libs1.0/lef/c.lef
../libs/lef/d.lef}

read_netlist design.v

init_design
```

Where mmmc.tcl has:

```
create_library_set -name test -timing
{/usa/db1/libs1.0/mmmc/a.lib
../libs/mmmc/b.lib }
```

Then after `write_db`, without `-user_path`, will save the libs and lefs to link to the real files.

```
> write_db db1.2
```

The ./mmmc.tcl file is always stored inside the DB dir, so no external reference is used.

By default, external file references are resolved to their fullpath name without any symlinks.

The .lib files inside the mmmc.tcl file will be referenced as:

```
{/global/project1/libs/mmmc/a.lib

/global/project1/libs/mmmc/b.lib}
```

The .lef files will be referenced as:

```
{/global/project1/libs/lef/c.lef
 /global/project1/libs/lef/d.lef}
```

This is portable and can be loaded by read_db anywhere on the same network.

```
/usa/db1.2/libs1.2/mmmc/a.lib    →
/global/project1/libs/mmmc/a.lib

/usa/db1.2/libs1.2/mmmc/b.lib    →
/global/project1/libs/mmmc/b.lib

/usa/db1.2/libs1.2/lef/c.lef    →
/global/project1/libs/lef/c.lef

/usa/db1.2/libs1.2/lef/d.lef    →
/global/project1/libs/lef/d.lef
```

With `-user_path` will save the libs and lefs to keep the symlinks of the current db.

```
> write_db db1.2 -user_path
```

The .lib files will be referenced as entered by the user: `{/usa/db1/libs1.0/mmmc/a.lib ../libs/mmmc/b.lib}`

The .lef files will referenced as entered by the user: `{/usa/db1/libs1.0/lef/c.lef ../libs/lef/d.lef}`

So `read_db` must be run in the same directory where `write_db` was run, or in a directory with the same ./y and /x directory structure.

```
/usa/db1.2/libs1.2/mmmc/a.lib   →
/usa/db1/libs1.0/mmmc/a.lib

/usa/db1.2/libs1.2/mmmc/b.lib   → ../../../libs/mmmc/b.lib

/usa/db1.2/libs1.2/lef/c.lef
→ /usa/db1/libs1.0/lef/c.lef

/usa/db1.2/libs1.2/lef/d.lef
→ ../../../libs/lef/d.lef
```

`-verilog`	Saves a Verilog file inside the DB directory, in addition to the normal DB files. This parameter is not normally used for flat or block flows, and is not used by read_db. It can be used in a hierarchical flow, where you want to use assemble_design to read only the Verilog and the DEF files without opening up each design DB and writing out the Verilog and DEF files separately (see -def). *Data_type*: bool, optional

Example

- The following command writes the current design into directory `test.dat`:

  ```
  write_db test.dat
  ```

- If you already ran timing analysis, you can save some run time during read like this:

  ```
  report_timing #first time usage, builds timing-graph
  write_db test.dat -timing_graph
  ```

  Then in a new session

  ```
  read_db test.dat
  report_timing #reports results, without repeating timing analysis
  ```

- For physical-only editing, you can do this to speedup read_db/write_db by ignoring the timing during read_db, and adding it back during write_db.

  ```
  read_db test1.dat -no_timing
  <physical-only editing>
  write_db test2.dat -add_ignored_timing
  ```

  Then in a new session

  ```
  read_db test2.dat #has timing from test1.dat
  ```

  If the logical netlist was modified during editing of test1.dat, then reading test2.dat may have unpredictable timing results.

Related Information

- `assemble_design`
- `read_db`
- `read_netlist`
- `set_multi_cpu_usage`
- `write_ldb`
- Design Import and Export in Stylus chapter in the Innovus Stylus Common UI User Guide.
- Partitioning the Design chapter in the Innovus Stylus Common UI User Guide.
- Innovus Stylus Common UI Mixed Signal (MS) Interoperability Guide for OpenAccess usage.

write_db Category Attributes

The root attributes in the `write_db` category are:

- write_db_binary_timing_constraints
- write_db_create_read_file
- write_db_include_metal_fill_rules
- write_db_copy_timing_constraints_always
- write_db_save_unused_lef_block_names

To set the value of any root attribute, use the `set_db` command:

```
set_db attribute_name value
```

To get the current value of any root attribute, use the `get_db` command:

```
get_db attribute_name
```

To get the current value of all attributes in the `write_db` category, use the following command:

```
get_db -category write_db
```

write_db_binary_timing_constraints

```
write_db_binary_timing_constraints {true | false}
```

Default: `true`

Data_type: bool, read/write

If set to `true`, `write_db` will save timing constraints in binary format. The value of the global should be persistent in subsequent `write_db` and `read_db` operations.

write_db_create_read_file

```
write_db_create_read_file {true | false}
```

Default: `false`

Data_type: bool, read/write

When set to `true`, `write_db my_db` will also create a '`my_db.read`' file with a '`read_db my_db`' command inside it. This is a Tcl file that can be passed to the Linux executable to make it easy to load the DB at startup like this: '`innovus -files my_db.read`'.

For OA usage, it would have the appropriate '`read_db -oa_lib_cell_view {<lib> <cell> <view>}`' command inside it.

When set to `false`, it only saves the design data (‘`my_db`’). User can read design as follows:

```
read_db my_db (innovus data)
read_db -oa_lib_cell_view {<lib> <cell> <view>} (OA data)
```

write_db_include_metal_fill_rules

```
write_db_include_metal_fill_rules {true | false}
```

Default: `false`

Data_type: bool, read/write

Specifies whether metal fill settings are to be saved in the .init file when saving the design. Metal fill settings include `set_metal_fill`, `set_metal_fill_spacing_table`, and `set_via_fill` settings. By default, these metal fill settings are lost when you exit Innovus. To enable Innovus to record metal fill settings in the .init file, you must set the `write_db_include_metal_fill_rules` to `1` before saving the design. You can easily reload your settings by restoring the saved design:

```
set write_db_include_metal_fill_rules true

set_metal_fill -iterationName iter1 ...

write_db

read_db

#The previous metal fill settings are reloaded.
```

Related Commands

- `set_metal_fill`
- `set_metal_fill_spacing_table`
- `set_via_fill`
- `write_db`
- `read_db`

write_db_copy_timing_constraints_always

```
write_db_copy_timing_constraints_always {true | false}
```

Default: `false`

Data_type: bool, read/write

Specifies whether SDC files are always copied to the DB directory rather than links to external SDC files. By default they are only copied if the constraints were modified.

This is the equivalent to always using the `-sdc` option whenever `write_db` is called.

Related Commands

- `write_db`
- `read_db`

write_db_save_unused_lef_block_names

```
write_db_save_unused_lef_block_names {true | false}
```

Default: `false`

Data_type: bool, read/write

Specifies whether to remember a list of unused LEF block macros (CLASS BLOCK or RING) so they are not read back with `read_db`. If you have a large number of LEF blocks (for example, memories) that are not used in this design, this can speedup `read_db` and reduce memory by skipping those LEF cells. Small blocks with less then 10 pins are still read in, because they are sometimes added later in the flow.

write_def

```
write_def
[-help]
def_file
[-add_half_wire_extension_on_pin]
[-all_layers]
[-bump_as_pin]
[-cut_row]
[-flatten_via_pillar]
[-floorplan]
[-io_row]
[-netlist]
[-no_core_cells]
[-no_special_net]
[-no_std_cells]
[-no_tracks]
[-no_virtual_trim]
[-output_mask_layers layerNames]
[-scan_chain]
[-selected]
[-skip_trim_metal_layers]
[-special_route_user_class user_class]
[-trim_metal_gap_fill]
[-unit unit_per_micron]
[-unplaced]
[-use_eeq_cell_with_liberty_info]
[-used_via]
[-verilog_from_def_netlist_flow]
[-with_shield]
[-routing  [-routing_to_special_net  | -wrong_way_routing_to_special_net ]]
[-early_global_route]
```

Writes the specified information to a DEF file. By default, the `write_def` command writes floorplan, and all placed and unplaced standard cell information to the DEF file.

The `write_def` command does not support multiple groups per region. If you run Early Global Route before you run the `write_def` command, the software does not write information about `FIXED` status wires to the DEF file. The `write_def` command supports rectilinear die area.

By default, the `write_def` command does not write out any `NONDEFAULTRULE` or `VIA` defined in the LEF file. To write out a LEF via, set the `write_def_include_lef_vias` attribute to `true`. To write out a nondefault rule, set the `write_def_include_lef_ndr` attribute to `true`.

You can use the `write_def` command after importing a design.

Note: You can select the DEF version for your output file by setting the `write_def_lef_out_version` attribute to the version number. For example, to set the version number to 5.6, type the following command:

```
set_db write_def_lef_out_version 5.6
```

Parameters

`-help`	Prints a brief description that includes the type and default information for each `write_def` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man write_def`.
`-add_half_wire_extension_on_pin`	Typically, signal routing in the `NETS` section uses the default half-width wire extensions everywhere except for zero extensions on pins. This parameter forces the zero-extension on pins to the default half-width extension, and makes the DEF compatible for older tools that do not support DEF zero wire extensions. **Note:** Using this parameter can cause DRC violations. *Data_type*: bool, optional
`-all_layers`	Writes out any wires and shapes on non-routing layers to SPECIALNETS section. *Data_type*: bool, optional
`-bump_as_pin`	Writes bump cells as top-level pins in the `PINS` section. If a PORT in a block has a PASSIVATION layer, the tool treats it as an embedded bump. `write_def -bump_as_pin` outputs all the layer information for embedded bumps in the PIN section as for normal bumps. Use the `-bump_as_pin` parameter to generate a DEF file for use with Quantus gate-level extraction. A DEF file generated with this parameter cannot be read back into the Innovus software. *Data_type*: bool, optional
`-cut_row`	Cuts rows against placement obstructions, block halos, or power domain fence minimum gaps. *Default:* Placement blockages are written to the DEF file. *Data_type*: bool, optional
def_file	Specifies the DEF output file. If you add `.gz` as a suffix, the `write_def` command will automatically compress the generated file. *Data_type*: string, required

`-flatten_via_pillar`	Some advanced nodes that use SADP (self-aligned double patterning) use via-pillars. A via-pillar is a single via definition with two or more disjoint metal-shapes on the bottom (or top) routing layer of the via. This option will flatten the via pillar into several separate "normal" vias (e.g. a via with only a single metal shape on the top and bottom via layers). The multiple normal vias will be geometrically equivalent to the single via pillar. This can be useful for external tools that do not support a via pillar. **Note:** DEF created with this option should not be read back into Innovus. It should only be used for external tools that do not support via-pillars properly. *Data_type*: bool, optional
`-floorplan`	Writes the floorplan data to the DEF file. This information includes rows, chip size, `CLASS PAD` instances, `CLASS BLOCK` instances, and all placed standard cells. *Data_type*: bool, optional
`-io_row`	Generates I/O rows based on I/O pad placement, and writes the I/O row information to the DEF file. *Data_type*: bool, optional
`-netlist`	Writes the netlist (that is, the routing connectivity information) to the DEF file. This option is typically used after the design has been placed. **Note:** Use `-netlist` with `-unplaced` if the design has not been placed. *Data_type*: bool, optional
`-no_core_cells`	Excludes core cell information from the DEF file to reduce the data size for import into APD. *Data_type*: bool, optional
`-no_special_net`	Excludes special vias, special routes, and power from the DEF file. *Data_type*: bool, optional
`-no_std_cells`	Excludes placed and unplaced standard cell information from the DEF file. **Note:** If you specify this parameter with the `-floorplan` or `-unplaced` parameters, it overrides their setting. *Data_type*: bool, optional
`-no_tracks`	Excludes output track and gcell statements to the DEF file. *Data_type*: bool, optional
`-no_virtual_trim`	Excludes the trim metal shapes on a virtual trim metal layer. *Data_type*: bool, optional

`-output_mask_layers` *layerNames*	Outputs MASK data on specified layers only. *Data_type*: string, optional
`-routing`	Writes the routing information to the `NETS` section of the DEF file. If the `-routing` option is not specified, both power and signal vias are not written to the DEF file. **Note:** The `-routing` parameter implies the `-netlist` parameter; therefore, if you specify this parameter, you do not need to specify `-netlist`. *Data_type*: bool, optional
`-routing_to_special_net`	Writes all regular NETS routing to the SPECIALNETS section. *Data_type*: bool, optional
`-scan_chain`	Writes scan chain information to the DEF file. This option is mandatory to ensure that START/STOP instances of scan chain are preserved during timing/power closure when reloading the DEF in another session. *Data_type*: bool, optional
`-selected`	Writes out information for selected as follows: • If an instance or net is selected, writes only the DEF header and COMPONENT section for instances you have selected in the Innovus main window. If you specify the `-routing` parameter along with this parameter, the `write_def` command writes selected nets with their special routing and regular routing to the NETS section of the DEF file. • If a net is not selected but individual segments or vias are selected, writes only the selected vias to the DEF file. All LEF vias are excluded by default. If the `write_def_include_lef_vias` attribute is set to `true`, `write_def -selected` includes selected LEF vias when writing out selected vias. If the `-no_special_net` option is specified, special vias on the selected net will not be written out. • If a bump is selected, writes bump pin connection to the `SPECIALNETS` section of the DEF file. If a net is selected but bump pins are not individually selected, bump pin connection is not written. • If a blockage is selected in the main window, writes blockage information to the DEF file. • If a group is selected, all the instances in the selected groups are treated as selected and are writtten out. *Data_type*: bool, optional

`-skip_trim_metal_layers`	Specifies to not write trimmetal shapes in the FILLS section. *Data_type*: bool, optional
`-special_route_user_class`	Specifies the user class for special route. *Data_type*: string, optional
`-early_global_route`	Writes information about the wires generated by Early Global Router to the `NETS` section of the DEF file. This option implies the `-routing` parameter; therefore, if you specify this parameter, you do not need to specify `-routing`. *Data_type*: bool, optional
`-trim_metal_gap_fill`	Writes trimmetal gap fills to a dummy net _TRIMMETAL_FILLS_RESERVERD in SPECIALNETS section. *Data_type*: bool, optional
`-unit` *unit_per_micron*	Defines the units for the DEF file. *Data_type*: int, optional
`-unplaced`	Writes information about unplaced standard cells to the DEF file. *Data_type*: bool, optional
`-use_eeq_cell_with_liberty_info`	Specifies to use the EEQ cell with the timing library information as the master cell of a leaf instance. *Data_type*: bool, optional

`-used_via`	Specifies that only used vias should be written to the DEF file. Unused vias are ignored. All LEF vias are also excluded by default. To include used LEF vias in the DEF file, set the `write_def_include_lef_vias` attribute is set to `true` before specifying `write_def -used_via`. • If the `-routing` option is specified along with `-used_via`, used signal vias are also written out. • If the `-no_special_net` option is specified along with `-used_via`, used special vias are not written out. • If you specify both `-used_via` and `-selected`, the `-used_via` option is ignored and only selected vias are written to the DEF file. The following warning message is issued: `**WARN: (ENCDF-1039): Option -used_via will be ignored because option -selected is specified. Only selected vias will be output.` • If neither `-selected` nor `-used_via` is specified and the `write_def_include_lef_vias` attribute is set to `false` (default), all DEF vias are written out. • If neither `-selected` nor `-used_via` is specified and the `write_def_include_lef_vias` attribute is set to `true`, all DEF and LEF vias are written out. *Data_type*: bool, optional
`-verilog_from_def_netlist_flow`	
	Reconciles the names in the DEF and Verilog files generated by the `read_netlist -def` command. The `read_netlist -def` command generates a flat Verilog file in which all of the special characters are escaped. Use this parameter to convert the escaped names back to the original names in the DEF file. *Data_type*: bool, optional
`-with_shield`	By default, `write_def` does not write out the shielding routing on the power/ground net when outputting regular routing of selected nets in NETS section. When `-with_shield` is specified, `write_def` writes out those shielding routing (if any) on the power/ground net which is in the SPECIALNETS section. The `-with_shield` option must be used with the `-routing` and `-selected` options. **Note:** Multiple nets can be selected. If none of the selected nets have shielding routing, the `-with_shield` option has no effect. *Data_type*: bool, optional

`-wrong_way_routing_to_special_net`	Writes wrong way routing in the SPECIALNETS section. *Data_type*: bool, optional

Example

- The following command writes floorplanning data, and information on placed and unplaced standard cells to `TOPCHIP_SP.def`:

  ```
  write_def -floorplan -unplaced TOPCHIP_SP.def
  ```

- The following command also writes floorplanning data, and information on placed and unplaced standard cells:

  ```
  write_def test.def
  ```

- The following command writes floorplan and standard cell information to the DEF file. It does not write out unplaced standard cell information.

  ```
  write_def -floorplan
  ```

- The following command excludes all information on placed and unplaced standard cells from the DEF file:

  ```
  write_def -floorplan -no_std_cells
  ```

  The resulting DEF file contains floorplan data only.

- The following command writes the shield wires that are shielding the selected nets to the DEF file:

  ```
  write_def -selected -routing -with_shield test.def
  ```

- The following command writes bump cells as top-level pins in the PINS section of the `test.def` file:

  ```
  write_def -bump_as_pin test.def
  ```

  The PINS section in the test.def file is as follows:

  ```
  ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
  PINS 102 ;
  - iot_schm_2m16m_3v + NET iot_schm_2m16m_3v + DIRECTION OUTPUT + USE SIGNAL
  + SPECIAL + POLYGON ap ( 93300 65970 ) ( 65970 65970 )
  ( 65970 93300 ) ( 27330 93300 ) ( 27330 65970 ) ( 0 65970 )
  ( 0 27330 ) ( 27330 27330 ) ( 27330 0 ) ( 65970 0 )
  ( 65970 27330 ) ( 93300 27330 ) + FIXED ( 386690 2658905 ) N ;
  - tck + NET tck + DIRECTION INPUT + USE SIGNAL
  + SPECIAL + POLYGON ap ( 93300 65970 ) ( 65970 65970 )
  ( 65970 93300 ) ( 27330 93300 ) ( 27330 65970 ) ( 0 65970 )
  ```

```
( 0 27330 ) ( 27330 27330 ) ( 27330 0 ) ( 65970 0 )
( 65970 27330 ) ( 93300 27330 ) + FIXED ( 53350 2658905 ) N ;
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

Related Information

- write_def Category Attributes
- Design Import and Export in Stylus chapter in the *Innovus Stylus Common UI User Guide*
- Using the NanoRoute Router chapter in the *Innovus Stylus Common UI User Guide*

write_def_by_section

```
write_def_by_section
[-help]
def_file out_filename
[-add_half_wire_extension_on_pin]
[-bump_as_pin]
[-comp_placement]
[-early_global_route]
[-fills]
[-groups [-no_group_regex]]
[-io_row]
[-net_routing]
[-no_comp]
[-no_core_cells]
[-non_default_rules]
[-no_die_area]
[-no_nets]
[-no_virtual_trim]
[-no_wildcard]
[-output_mask_layers OutputMaskData]
[-pins]
[-place_blockages]
[-route_blockages]
[-rows]
[-scan_chains]
[-selected]
[-special_net_routing]
[-special_nets]
[-special_route_user_class user_class]
[-special_shape swire_type_list]
[-tracks]
[-unit unit_per_micron]
[-unplaced]
[-used_via]
[-verilog_from_def_netlist_flow]
[-vias]
```

Writes information to specified sections in a DEF file. By default, it writes DIEAREA (and associated FE_CORE_BOX* properties), and the netlist part (not the placement or routing part) of the COMPONENTS and NETS sections that must be explicitly suppressed with options below if you do not want them. Other sections or sub-sections are only added based on option settings.

By default, the `write_def_by_section` command does not write out any via defined in the LEF file. If the `write_def_include_lef_vias attribute` is set to `1`, the `write_def_by_section` command writes out the via defined in LEF file.

You can use the `write_def_by_section` command after importing a design.

Parameters

`-help`	Prints a brief description that includes the type and default information for each `write_def_by_section` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man write_def_by_section`
`-add_half_wire_extension_on_pin`	Forces the zero-extension on pins to the default half-width extension.
`-early_global_route`	Writes information about the wires generated by Early Global Route to the `NETS` section of the DEF file.
`-bump_as_pin`	Writes bump cells as top-level pins in the PINS section. Use this parameter to generate a DEF file for use with Quantus gate-level extraction. **Important**: A DEF file generated with this parameter cannot be read back into the Innovus software.
`-comp_placement`	Writes component placement and orientation information to the `COMPONENTS` section of the DEF file. The output is limited to instances with PLACED/FIXED/COVER status, unless `-unplaced` is also specified.
`def_file` *out_filename*	Specifies the name of the DEF output file.
`-fills`	Writes metal fill information to the `FILLS` section of the DEF file.
`-groups`	Writes group information to the `GROUPS` section of the DEF file. When you use this parameter, Innovus also writes information about regions.
`-io_row`	Generates I/O rows based on I/O pad placement and writes the I/O row information to the DEF file.
`-net_routing`	Writes routing data to the `NETS` section of the DEF file.
`-no_comp`	Excludes component information from the DEF file.
`-no_core_cells`	Excludes core cell information from the DEF file to reduce the data size for import into APD.

`-non_default_rules`	Writes non default rules (NDRs) to the NONDEFAULTRULES section of the DEF file. **Note**: The `-non_default_rules` parameter works only with `-no_nets`. As the NETS section is always dumped by default, so the NONDEFAULTRULE section is also dumped by default. However, if `-no_nets` is specified, the NONDEFAULTRULE section will be dumped only if -non_default_rules is also specified.
`-no_die_area`	Excludes the DIEAREA and associated FE_CORE_BOX* properties (FE_CORE_BOX_LL_X, FE_CORE_BOX_UR_X, FE_CORE_BOX_LL_Y, FE_CORE_BOX_UR_Y). You can use this option if you do not need some sections, such as SCANCHAINS or NONDEFAULTRULES, to be consistent with the DIEAREA or core-box settings.
`-no_nets`	Excludes net information from the DEF file. **Note**: If `-no_nets` is specified, the `-net_routing` and `-early_global_route` parameters are ignored.
`-no_group_regex`	Suppresses the use of regular expressions to represent instance names. You must also specify the `-groups` parameter.
`-no_virtual_trim`	Excludes the trim metal shapes on a virtual trim metal layer. *Data_type*: bool, optional
`-no_wildcard`	Suppresses the use of the wildcard format to represent special net connectivity. If specified, the `-compPlacement` parameter is ignored. **Note**: The `-no_wildcard` parameter must be used along with the `-special_nets` parameter to be effective. If you specify `-no_wildcard` without `-special_nets`, the software ignores it. *Default:* If you do not specify this parameter, Innovus writes connectivity information in the `SPECIALNETS` section in wildcard format, if possible.
`-output_mask_layers` *OutputMaskData*	Specifies the output MASK data on specified layers only.
`-pins`	Writes pin information to the `PINS` section of the DEF file.
`-place_blockages`	Writes placement blockage information to the `BLOCKAGES` section of the DEF file.
`-route_blockages`	Writes routing blockage information to the `BLOCKAGES` section of the DEF file.

`-rows`	Writes row information to the `ROWS` section of the DEF file.
`-scan_chains`	Writes scan chain information to the `SCANCHAINS` section of the DEF file.
`-selected`	Writes out information for selected as follows: • If a net or instance is selected, writes only the DEF header and `COMPONENT` section for instances or nets you have selected in the Innovus main window. If you specify the `-special_net_routing` parameter along with this parameter, the `write_def_by_section` command writes selected nets with their special routing to the NETS section of the DEF file. • If a net is not selected but individual segments or vias are selected, writes only the selected vias to the DEF file. All LEF vias are excluded by default. If the `write_def_include_lef_vias` attribute is set to 1, `write_def_by_section -selected` includes selected LEF vias when writing out selected vias. • If a bump is selected, writes bump pin connection to the `SPECIALNETS` section of the DEF file. If a net is selected but bump pins are not individually selected, bump pin connection is not written. • If a blockage is selected in the main window, writes blockage information to the DEF file. • If a group is selected, all the instances in the selected groups are treated as selected and are writtten out.
`-special_nets`	Writes the pin connections for special nets to the `SPECIALNETS` section of the DEF file. When specified along with `-selected`, writes out the pin connections for only selected nets. Does not write routing information.

`-special_net_routing`	Writes routing information for special nets to the `SPECIALNETS` section of the DEF file, when specified with the `-special_nets` parameter. Writes only the routes, without the pin connectivity information, to the `SPECIALNETS` section when specified without the `-special_nets` parameter. Use this parameter with `-selected` to write routing information of only selected nets. Use this parameter with `-special_route_user_class` to write routing information of only those special routes that have a `user_class` attribute from the specified list.
`-special_route_user_class` *`user_class`*	
	Writes out only the user class for special route.
`-special_shape` *`swire_type_list`*	Writes special routing information with specified shape types. **Note:** Use `write_def_by_section -special_shape` after you have used the `set_special_route_type` command to mark special wires and vias and set them to a specific shape type.
`-tracks`	Writes routing track information to the `TRACKS` and `GCELLGRID` statements of the DEF file.
`-unit unit_per_micron`	Defines the units for the DEF file.
`-unplaced`	Includes UNPLACED instances in the `COMPONENTS` section when combined with the `-compPlacement` option.

`-used_via`	Specifies that only used vias should be written to the DEF file. Unused vias are ignored. All LEF vias are also excluded by default. To include used LEF vias in the DEF file, set the `write_def_include_lef_vias` attribute to `1` before specifying `write_def_by_section -used_via`. • If you specify both `-used_via` and `-selected`, the `-used_via` option is ignored and only selected vias are written to the DEF file. The following warning message is issued: **WARN: (ENCDF-1039): Option `-used_via` will be ignored because option `-selected` is specified. Only selected vias will be output. • If neither `-selected` nor `-used_via` is specified and the `write_def_include_lef_vias` attribute is set to `0` (default), all DEF vias are written out. • If neither `-selected` nor `-used_via` is specified and the `write_def_include_lef_vias` attribute is set to `1`, all DEF and LEF vias are written out.
`-verilog_from_def_netlist_flow`	
	Reconciles the names in the DEF and Verilog files generated by the loadDefFile command. The loadDefFile command generates a flat Verilog file in which all of the special characters are escaped. Use this parameter to convert the escaped names back to the original names in the DEF file.
`-vias`	Writes via information to the `VIAS` section of the DEF file.

Example

- The following command writes component placement, pin, and special net information to the `test.def` file:

  ```
  write_def_by_section -comp_placement -pins -special_nets test.def
  ```

- In the following set of commands, `set_special_route_type` first traces special wires and vias that touch the named bump cells and sets them to `IOWIRE`. `write_def_by_section` then writes the special routing with IOWIRE shape type to the `test1.def` file.

  ```
  set_special_route_type -bump_cell {bc*} -stop_type_list {IOWIRE} -type IOWIRE

  write_def_by_section -special_shape {IOWIRE} test1.def
  ```

- With only the VDD net selected in the main window, the following command writes the routing information without pin connections for only those special routes in VDD that have the user_class attribute name as `attrA`.

  ```
  write_def_by_section -special_net_routing -special_route_user_class {attrA} -selected
  ```

Related Topics

- Placing the Design chapter of the *Innovus Stylus Common UI User Guide*
  - Saving Scan Files

write_def Category Attributes

The root attributes in the write_def category are:

- write_def_hierarchy_delimiter
- write_def_compress_vias
- write_def_include_lef_ndr
- write_def_include_lef_vias
- write_def_polygon_die_area
- write_def_lef_out_version

To set the value of any root attribute, use the `set_db` command:

```
set_db attribute_name value
```

To get the current value of any root attribute, use the `get_db` command:

```
get_db attribute_name
```

To get the current value of all attributes in the `write_def` category, use the following command:

```
get_db -category write_def
```

write_def_hierarchy_delimiter

`write_def_hierarchy_delimiter` *`delimiter`*

Default: `NULL`

Data_type: string, read/write

Specifies the hierarchy delimiter character for the `write_def` and `write_def_by_section` commands.

For example, to set / as the delimiter, use the following code:

```
set write_def_hierarchy_delimiter /
```

Note:

- You cannot unset this attribute.
- `write_def_hierarchy_delimiter` is for the `write_def` and `write_def_by_section` commands only. It does not change the hierarchy delimiter character of Innovus.

The hierarchy delimiter used appears in the output DEF File as shown below.

```
DIVIDERCHAR "/" ;
```

Related Command

- write_def
- write_def_by_section

write_def_compress_vias

```
write_def_compress_vias number
```

Default: 0

Data_type: int, read/write

Compresses the via statement in SPECIALNETS section. This attribute controls write_def to sort vias with two dimension array, so it can use DO numX BY numY STEP stepX stepY statement.

The via array information is only preserved in the read_def session, but it will not be kept by write_db/read_db.

Related Command

- write_def

write_def_include_lef_ndr

```
write_def_include_lef_ndr {true | false}
```

Default: false

Data_type: bool, read/write

Controls whether write_def writes out the LEF non-default rule information in DEF. By default, write_def does not write out non-default rules defined in LEF. Set this attribute to true to output non-default rules defined in LEF.

Related Command

- write_def
- write_def_by_section

write_def_include_lef_vias

```
write_def_include_lef_vias {true | false}
```

Default: false

Data_type: bool, read/write

Controls whether `write_def` writes out LEF via information in DEF. By default, `write_def` does not write out vias defined in LEF. Set `write_def_include_lef_vias` to `true` to write out LEF vias in the DEF file.

Related Command

- `write_def`
- `write_def_by_section`

write_def_polygon_die_area

```
write_def_polygon_die_area {true | false}
```

Default: `true`

Data_type: bool, read/write

By default, `write_def` writes out polygon DIEAREA if the design is rectilinear. To write out rectangular DIEAERA along with blockages in cut-out area for a rectilinear design, set this attribute to `false`.

Related Command

- `write_def`

write_def_lef_out_version

```
write_def_lef_out_version {5.5 5.6 5.7 5.8}
```

Default: `5.8`

Data_type: enum {5.5 5.6 5.7 5.8}, read/write

Specifies LEF/DEF output version. Possible string values are `5.6, 5.7, and 5.8`. The default is `5.8`.

write_design

```
write_design
[-help]
directory_name
[-force_sdc]
[-pg_netlist]
[-prefix prefix_name]
[-rc_extract]
[-categories category_list]
[-signoff_eco]
-tempus
```

Exports all collateral of a design that is necessary to reconstitute the design intent in Tempus through import of discrete data files. These data files are normally gzipped, ASCII text files.

The write_design command exports the following data:

- The flat Verilog netlist for the design - optionally with power/ground
- The complete DEF for the design
- The CPF or UPF power intent
- Metrics data from Innovus
- Common preserve data
- Global and mode-level constraint data
- An MMMC configuration organizing the relevant data and establishing the timing intent
- SPEF RC parasitic files

The `write_design` command will generate the following template Tempus initialization scripts:

`designName.tempus_init.tcl`

`designName.tempus_setup.tcl`

These scripts will illustrate how to start a standard Tempus STA session using the data provided.

If the `-signoff_eco` option is also specified, then the following additional scripts will be generated:

`designName.tempus_eco_init.tcl`

`designName.tempus_eco_setup.tcl`

The `*init.tcl` scripts just have the bare minimum to load the netlist up through init_design. The `*setup.tcl` scripts include adding other side information, such metrics, modes, that might be helpful.

Parameters

`directory_name`	Specifies the name of the output directory to write the design files. *Data_type*: string, required
`-force_sdc`	The designName.mmmc.tcl file created by write_design includes references to the SDC constraint files. If the Innovus MMMC configuration is not pointing to golden SDCs, you can use this option to force the constraints to be written out locally to the write_design hierarchy. If the constraints have been modified during the Innovus session, the SDCs will automatically be saved locally. *Data_type*: bool, optional
`-help`	Outputs a brief description that includes type and default information for each `write_design` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man write_design`.
`-pg_netlist`	By default, write_design exports a logical Verilog netlist. When this option is specified, the netlist will include additional PG pin and PG net information. You should use this option when the Tempus PGV flow is used to derive power intent directly from the netlist - rather than using CPF or UPF. *Data_type*: bool, optional
`-prefix` *prefix_name*	By default, files written to the output directory are prefixed by the design name (e.g. myChip.mmmc.tcl). You can use the -prefix option to specify an alternate prefix name. *Data_type*: string, optional

`-rc_extract`	Enables the output of **SPEF** formatted RC extraction data for each active rc_corner. write_design will only export parasitics that are currently available in the system. It does not invoke extraction if parasitic data is missing or try to configure/invoke sign-off level extraction. If the -rc_extract option is provided and parasitics are absent in the system, the tool will give a Warning or an Error. **Note**: RC data will not be saved if the design has not been extracted or the extract_rc_engine setting is 'pre_route'. *Data_type*: bool, optional
`-categories` *`category_list`*	By default, write_design will save key init settings to the Tempus template scripts from the current Innovus session. You can select additional settings belonging to various get_db attribute categories, such as timing, init, delay_cal, to be exported to a separate file for reference. *Data_type*: string, optional
`-signoff_eco`	Writes additional data (including DEF) to support Tempus physically-aware sign-off ECO flows. The ECO template script will also include the commands and file pointers for loading DEF and LEF. *Data_type*: bool, optional
`-tempus`	Saves the output for Tempus usage. *Data_type*: bool, required

Related Information

- `write_lec_dft_constraints attribute`
- Design Import and Export in Stylus chapter in the *Innovus Stylus Common UI User Guide*

write_do_lec

```
write_do_lec
[-help]
[output_filename | > output_filename]
[-checkpoint lec_checkpoint_file]
[-golden_design filename]
[-log_file filename]
[-no_exit]
[-pre_compare filename]
[-pre_exit filename]
[-pre_read filename]
[-revised_design filename]
[-verbose]
[-write_session lec_session_name]
[-hier | -flat]
```

Writes a script that can be used by Conformal® Logical Equivalence Checking tool to compare the specified golden and revised designs.

For formal verification of the Innovus-generated netlists, the `write_do_lec` command generates, within Innovus, the Conformal LEC scripts (such as dofile). These scripts are used for equivalence checking. You can generate a flat or a hierarchical comparison script. The dofile written out by this command contains standard Conformal flow commands and you will need to add the DFT constraints if netlist is scan inserted.

The `write_do_lec` command creates a verification directory, if does not already exist, in which the name mapping information file will be created. The name mapping file will usually contain the mapping information of MBIT flops against its split/single bit candidate. By default, the name of this verification directory is `fv/invs/%s`. You can use the `write_lec_directory_naming_style` attribute to change the directory name.

Parameters

`-help`	Prints a brief description that includes the type and default information for each `write_do_lec` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man write_do_lec`
`-checkpoint lec_checkpoint_file`	Specifies the name of the LEC checkpoint file to save at the end. *Data_type*: string, optional
output_filename	Specifies the name for the output file to be written. *Data_type*: string, required

`-flat`	Creates a flat comparison script. *Data_type*: bool, optional
`-golden_design` *filename*	Specifies the user-written netlist file name to use as the golden design. *Data_type*: string, optional
`-hier`	Creates a hierarchical comparison script. *Data_type*: bool, optional
`-log_file` *filename*	Specifies the name of the LEC log file. *Data_type*: string, optional
`-no_exit`	Skips the exit command at end of dofile. *Data_type*: bool, optional
`-pre_compare` *filename*	Specifies an extra dofile to be sourced right before 'compare'. *Data_type*: string, optional
`-pre_exit` *filename*	Specifies an extra dofile to be sourced right before 'exit'. *Data_type*: string, optional
`-pre_read` *filename*	Specifies an extra dofile to be sourced right before reading library/design. *Data_type*: bool, optional
`-revised_design` *filename*	Specifies the user-written netlist file name to use as the revised design. *Data_type*: string, optional
`-write_session` *lec_session_name*	Specifies the name of the LEC session to save at the end. *Data_type*: string, optional
`-verbose`	Specifies whether the generated dofile will report verbosely. *Data_type*: bool, optional
`>` *output_filename*	Specifies the name for the output file to be written. *Data_type*: redirect_operator, optional

Example

The following command writes a script that can be used by Conformal® Logical Equivalence Checking tool to compare the `synth.v` golden design with the revised designs.

```
write_do_lec \
    -flat \
```

```
-golden_design synth.v \
-log_file lec.log \
-pre_compare dft_constraints.do \
lec.do
```

Related Information

- `init_read_netlist_files attribute`
- `write_lec_directory_naming_style attribute`

write_lef_abstract

```
write_lef_abstract
[-help]
fileName
[-add_obs_layers layer_list]
[-cut_obs_min_spacing]
[-cut_obs_to_expose_routing distance_in_microns]
[-exclude_obs_layers layer_list]
[-exclude_pin_layers layer_list]
[-design_boundary point_list]
[-extract_block_obs]
[-extract_block_pg_pin_layers layer_number_list]
[-ignore_bump_on_pin {all pg signal}]
[-io_pad_pin]
[-no_cut_obs]
[-obs_spacing {val layers [val2 layers ...]} [-obs_spacing_per_layer ]]
[-property]
[-top_layer layer_number [-obs_above_top_layer layer+]]]
[-stripe_pins [-selected ]]
[-stripe_pins [-port_for_each_stripe_pin]]
[-stripe_pins [-pg_pin_layers layer_number_list]]
[-stripe_pins [-pg_nets net+]]
[-extract_well_layer_obs well_layer_list [-extract_well_obs_from_rows margin]]
[-extract_well_layer_obs well_layer_list [-merge_connected_well_only ]]
[ -5.6 | -5.7 | -5.8 | -6.0 ]
```

Generates design abstract (LEF) information for the current routed block-level design. By default, `write_lef_abstract` copies all the current design PG and signal pins to the LEF and creates cut-outs in the obstructions (OBS shapes) around the pins. The output LEF file contains process antenna information, if you run `check_process_antenna` before you run the `write_lef_abstract` command.

A routing blockage with a pushdown attribute (e.g. a DEF BLOCKAGE with + PUSHDOWN) is treated as an area of routing reserved for the level above this block. The LEF will have these blockages cut-out of the OBS. Other routing blockages shapes are ignored.

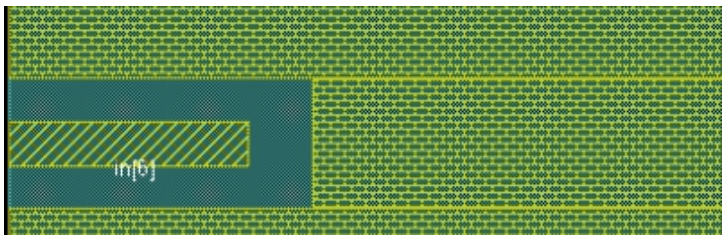
In a bottom-up flow (where the block was not created using the `create_partition` command), you normally create a simple LEF abstract that contains:

- one obstruction (`OBS`) shape per routing layer
- signal pins located along the block boundary
- add power pins for the top metal layer PG stripes (see –stripe_pins and –pg_pin_layers options)
- embedded bump pins shapes are also written out as LEF pin shapes

If the block used four routing layers out of a possible six layers, you would run the following command:

```
write_lef_abstract -no_cut_obs -stripe_pins -pg_pin_layers 4 -top_layer 4
```

Parameters

`-help`	Prints a brief description that includes the type and default information for each `write_lef_abstract` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man write_lef_abstract`
`-5.6 | -5.7 | -5.8 | -6.0`	Specifies the LEF version number. This value overrides any value you set with the `write_def_lef_out_version` attribute. *Default:* The software uses the current `write_def_lef_out_version` value. The default value for `write_def_lef_out_version` is `5.8`. *Data_type*: bool, optional
`-add_obs_layers` *layer_list*	Writes the shapes on on the specified layers as OBS shapes in the LEF. These layers cannot be routing or cut layers. *Data_type*: string, optional
`-cut_obs_min_spacing`	Creates cut outs in the blockages around pins using minimum width to calculate the spacing. This option cannot be used with `-no_cut_obs`. *Data_type*: bool, optional 
`-cut_obs_to_expose_routing` *distance_in_microns*	Exposes routing within the specified halo distance to a LEF model. *Data_type*: bool, optional

`-design_boundary` *point_list*	Specifies a list of points defining the design boundary polygon for the output LEF (a rectangle would use 4 points). It is used instead of the real design boundary to compute the LEF SIZE, OBS shapes to block routing layers, and OBS shapes on the OVERLAP layer if it is non-rectangular. The LEF will also include any special routes that are outside the design boundary as OBS shapes unless they are converted to PIN shapes based on other options. It does not matter if PIN shapes are inside or outside the design boundary, they are always written out. This is intended for a block that only has bumps and/or top-level RDL routing to the bumps that are outside the "real" block boundary. The real block boundary can be specified with this option to shrink the SIZE and OBS shapes to just include the active area of the block, and let the bumps and RDL routing stick outside. An important limitation is that any instances or regular routing shapes (for example, created by `route_design`) outside the design boundary will be ignored. It is assumed that placement and routing blockages are used to prevent this outside the "real" block boundary. *Data_type*: string, optional
`-exclude_obs_layers` *layer_list*	Ignores the specified layer when saving OBS in the macro LEF file. You can use this parameter to set the layer as the following format: ◦ One layer: For example, if you specify `-exclude_obs_layers {AP}`, the AP layer is not saved to OBS. ◦ Multiple layers: For example, if you specify `-exclude_obs_layers {M7 M8}`, M7, M8 layers are not saved to OBS. *Data_type*: (layer)+, optional
`-exclude_pin_layers` *layer_list*	Specifies not to create PIN shapes on the specified layers. *Data_type*: (layer)+, optional

`-extract_block_obs`	Extracts all obstruction (`OBS`) information from the instantiated `CLASS BLOCK`, `CLASS PAD AREAIO`, and `CLASS RING` macros and writes out the obstructions to the output LEF file. `OBS` information is extracted for layers that are not already being completely covered under the normal routing layer range and for cut layers between routing layers that are completely covered. *Data_type*: bool, optional
`-extract_block_pg_pin_layers` *layer_number_list*	

	Extracts the power and ground pin information from any `CLASS BLOCK`, `RING`, `PAD`, or `COVER` macros with pins on layers in the specified layer list and writes out the pin information to the LEF file.
	The layer names can be either names in the technology file like metal1, or technology independent names like 1 for the first routing layer, 2 for the second routing layer.
	For example, to create PG pins on layers `metal3` and `metal4`, use:
	`-extract_block_pg_pin_layers {metal3 metal4}`
	Or
	`-extract_block_pg_pin_layers {3 4}`
	Data_type: (layer)+, optional

`-extract_well_obs_from_rows` *margin*	Adds the bounding box of rows plus the *margin* to the bounding-boxes computed by -extract_well_layer_obs. This is useful if the std-cell LEFs do not have well shapes in them. The -extract_well_layer_obs option is required to use this option. *Data_type*: float, optional
`-extract_well_layer_obs` *well_layer_list*	Extracts the bounding-box of all the OBS and PIN shapes on all std-cells and blocks for each specified well layer. The bounding-box is trimmed to be inside the output LEF boundary and then written as a well-layer OBS shape to the output LEF file, This can be used for well spacing checks on the block at the next level if that is desired. *Data_type*: (layer)+, optional
fileName	Specifies the LEF output file. *Data_type*: string, required

`-ignore_bump_on_pin {all pg signal}`	By default, pin shapes of bump cells (e.g. LEF MACRO CLASS BUMP) are written out as LEF pin shapes. This option can be used to suppress those LEF pin shapes depending on the type of net they are connected to.
`-io_pad_pin`	Writes out the IO pads that are connected to the design pins as LEF pin shapes.
`-merge_connected_well_only`	Merges connected well shapes only, instead of merging all shapes into one bounding box. *Data_type*: bool, optional
`-no_cut_obs`	Creates a single obstruction shape for each routing layer. *Default:* Creates cut-outs (obstructions) in the blockages for signal pins, power and ground pins, vias, and power and ground stripes. *Data_type*: bool, optional This option cannot be used with `-cut_obs_min_spacing`.
`-obs_above_top_layer` *layer+*	Writes out OBS shapes rather than one large abstract OBS shape for special-routes on these layers that are not PIN shapes. The layers will have OBSSPACING FULLDRC added to the LEF so the OBS shapes on those layers are treated as 'real' shapes for DRC checking, and not abstract obstructions. This is intended for blocks that have some sparse special-routing above the blocks "normal" top-layer set with the `-top_layer` option (for example, it might be a few special-routes for clocks that match up with the chip clock distribution but must be visible inside the block). Only special-routes are currently supported. *Data_type*: layer+, optional

`-obs_spacing {val layers [val2 layers ...]}`	Writes out the LEF58_OBSSPACING property for the MACRO. This overrides the global USEMINSPACING OBS setting and sets the spacing required between routing and the OBS shapes of just this MACRO. This is equivalent to adding a SPACING *value* to every OBS RECT on the affected layers in the LEF. Note, it will only affect an OBS RECT that has no existing SPACING *value*.
	
	val = 'min' for min-spacing, and *layers* = {} for all layers.
	
	The order of the arguments is the order in the LEF file. In LEF, the last value for a given layer "wins" over previous values for that macro.
	
	For example, `-obs_spacing {min {} 1.0 {m2 m3} 2.0 {m5}}` would add this to the LEF:
	
	`PROPERTY LEF58_OBSSPACING "`
	`OBSSPACING MIN ;`
	`OBSSPACING 1.0 LAYER m2 LAYER m3 ;`
	`OBSSPACING 2.0 LAYER m5 ; " ;`
	
	So, OBS shapes in this MACRO on every layer would use SPACING <min-spacing> for that specific layer, then m2, m3 are changed to use SPACING 1.0, and finally m5 is changed to use SPACING 2.0.
	
	Data_type: string, optional

`-obs_spacing_per_layer`	Writes out the OBS SPACING values under each OBS LAYER statement rather than as a property that `-obs_spacing` uses by default. This parameter requires `-obs_spacing`.
	Data_type: bool, optional
	The following example sets the spacing of all layers as 1.0, except M2.
	`write_lef_abstract -obs_spacing {1.0 {} 2.0 {M2}} -obs_spacing_per_layer`
	`Macro A`
	`  ...`
	`  OBS`
	`    Layer M1` **SPACING 1.0** `;`
	`    RECT ... ;`
	`    Layer M2` **SPACING 2.0** `;`
	`    RECT ... ;`
	`    ...`
	`  END`
	`  ...`
	`END A`

`-property`	Writes out all the user-defined design properties.
`-pg_nets` *net+*	Writes out PG stripes only on these PG nets. If you specify a net name that does not exist in the design, or the net is not a PG net, tool will give an error and stop.
`-pg_pin_layers` *layer_number_list*	

	Writes out power and ground stripes on the specified layer numbers as power and ground pins.
	
	The layer names can be either names in the technology file like metal1, or technology independent names like 1 for the first routing layer, 2 for the second routing layer.
	
	For example, to create PG pins on layers metal3 and metal4, use:
	
	`-pg_pin_layers {metal3 metal4}`
	Or
	`-pg_pin_layers {3 4}`
	
	Note: You must specify the `-stripe_pins` parameter in order to use this parameter.
	
	Default: The top metal layer
	
	Data_type: (layer)+, optional

`-port_for_each_stripe_pin`	Creates a separate LEF port for each pin shape created by `-stripe_pins`. This means `check_connectivity` will check that every port is touching a PG routing shape at the next level. This is normally what users want for individual PG stripes on the top-layer of the block--each separate stripe shape should be connected to a top-level route or via somehow. This can help detect some missing power-mesh connections before running a complete IR drop analysis. *Data_type*: bool, optional
`-top_layer` *layer_number*	

	Creates obstructions (OBS shapes) covering the block only for layers up to the specified layer.
	
	The layer name can be either a name in the technology file like metal1, or a technology independent names like 1 for the first routing layer, 2 for the second routing layer etc.
	
	For example, to specify metal4 as the top layer, use:
	
	`-top_layer metal4`
	Or
	`-top_layer 4`
	
	By default, all routing layers in the technology file are fully obstructed with OBS shapes across the entire block (possibly cut around pin shapes). If this block does not use some of the highest routing layers, you should use this option to limit the OBS shapes to below layer.
	
	Data_type: (layer), optional

`-selected`	Writes out only the selected PG stripes as LEF pin shapes. *Data_type*: bool, optional
`-stripe_pins`	Writes out power and ground stripes on the top metal layer as power and ground pins. You can use this parameter in a bottom-up methodology, when you create abstracts for blocks that you will use at the next higher level of a hierarchical design. *Data_type*: bool, optional

Example

Suppose you have a design that was routed on only the first 5 routing layers, even though the technology has higher metal layers.

You would use the following command to include any PG stripes on metal5 as pins in the block LEF to allow connecting to them from above, and to only create OBS for routing layers up to layer 5.

```
write_lef_abstract my.lef -top_layer 5 -stripe_pins -pg_pin_layers 5
```

If your design also included a block inside that has PG pins on metal5 with no PG stripes in the current design covering them, you can include layer 5 PG pins from the internal blocks using the following command:

```
write_lef_abstract my.lef -top_layer 5 -stripe_pins -extract_block_pg_pin_layers 5
```

If you also want to include metal4 PG stripes as pins, and metal4 PG pins from the block, you would use the following command:

```
write_lef_abstract my.lef -top_layer 5 -stripe_pins -pg_pin_layers {4 5} -
extract_block_pg_pin_layers {4 5}
```

If you want to include some sparse special-route shapes on metal6, metal7 (for example, for a few manually routed clock nets) , you would use the following command:

```
write_lef_abstract my.lef -top_layer metal5  -obs_above_top_layer {metal6 metal7}
```

Related Information

- `create_partition`
- `check_connectivity`
- `check_process_antenna`
- `route_design`
- `write_def_lef_out_version` attribute.
- Design Import and Export in Stylus chapter in the *Innovus Stylus Common UI User Guide*
- Top-level Timing Closure Methodologies chapter in the *Innovus Stylus Common UI User Guide*
    - Example Top-Level Implementation Flow with ILMs

write_mmmc

```
write_mmmc
[-help]
[-library]
```

Writes out the MMMC file.

Parameters

`-help`	Prints out the command usage.
`-library`	Writes out only the library part of the MMMC file. *Data_type*: bool, optional

Related Information

- init_design
- `read_mmmc`
- read_physical
- `read_netlist`
- `set_analysis_view`

write_name_mapping

```
write_name_mapping
[-help]
[-bbox_pins]
[-hierarchical]
[-map_pins {input output all}]
[-pin_polarity]
[-enable_one_to_one_mapping | -report_power_format]
[-skip_clock_gate]
[-updated_only]
{output_filename | [[-output output_directory] -prefix filename_prefix]}
[-hdl | -initial | [-input_mapping_file mapping_file [-print_deleted]]]
```

Writes or updates the initial netlist key point names and corresponding current netlist key point names to a file. Mapping file consists of pin mapping information of all changes done during the flow (between comparison points) including polarity changes .

Parameters

`-help`	Prints a brief description that includes the type and default information for each `write_name_mapping` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man write_name_mapping`
`-bbox_pins`	Prints the pins of bbox in the output mapping file instead of inst names. • Prints bbox input pins if `-map_pins {input}` is specified. • Prints bbox output pins if `-map_pins {output}` is specified. • Prints both input and output pins if `-map_pins {all}` is specified.
`-enable_one_to_one_mapping`	When set to `true`, only one of the clones will be printed in the mapping file.
output_filename	Specifies the name of the output mapping file.
`-hdl`	Enables the generation of name mapping from current names to the names defined by the 'hdl_name' attribute. This will be used for power analysis.
`-hierarchical`	Specify if write out other post boundary optimizations mapping

`-initial`	Enables the generation of name mapping from current names to names defined by the 'initial_name' attribute. This will be used for verification of initial input netlist (post-synthesis) to any other netlist generated during the Innovus flow. The `write_do_lec` command in Innovus will call `write_name_mapping -initial` so that Conformal LEC do file and name mapping file are generated in one shot. The do file will source the name mapping file to facilitate key point mapping in LEC.
`-input_mapping_file` *`mapping_file`*	When provided Tool will read existing mapping file and write out updated mapping file.
`-map_pins {input output all}`	Specify type of pins to write out.
`-output` *`output_directory`*	Specify output directory
`-pin_polarity`	Specify if write out polarity information.
`-prefix` *`filename_prefix`*	Used for file name prefix.
`-print_deleted`	When `-print_deleted` is specified the mapping file will also print deleted instances.
`-report_power_format`	Specify if write out 1Pass power replay format
`-skip_clock_gate`	Filters out ICGs that are not in RTL. If you set `-skip_clock_gate` to `true` with `-hdl`, the tool skips the clock gate pins except the pins with hdl_name. If `-skip_clock_gate` is set to `true` and `-hdl` is not specified, the tool skips all clock gate pins.
`-updated_only`	Skips printing the pin that does not have any change/update during optimizations/flow.

Related Information

- `write_do_lec`
- `write_lec_directory_naming_style` attribute

write_netlist

```
write_netlist
file_name
[-help]
[-abstract]
[-apply_maskshift]
[-cell_instance_color {true | false}]
[-cell_master_color {true | false}]
[-derive_pg_port_dir]
[-exclude_cell_def cellName]
[-exclude_ilm]
[-exclude_leaf_cells | -only_leaf_cells | -only_std_cells | -only_blocks | -module moduleName]
[-exclude_insts_of_cells cellName]
[-exclude_top_pg_ports portName]
[-export_top_pg_nets]
[-follow_logical_module_order | -flat]
[-flatten_bus]
[-ignore_fixed_mask]
[-ilm]
[-include_bump_cells]
[-include_pg]
[-include_phys_cells cellName]
[-include_phys_insts]
[-keep_all_backslash]
[-line_length characters_per_line]
[-maskshift_suffix maskShiftSuffix]
[-omit_floating_ports]
[-phys]
[-remove_power_ground]
[-replace_inout_with_output_for_pg_ports]
[-replace_wor_wand_with_wire]
[-top_module cellName]
[-top_module_first]
[-update_tie_connections]
[-use_cds_thru_for_assign]
[-use_eeq_cell_map {masterCell EEQcell1 EEQcell2}]
[-use_eeq_cell_with_liberty_info]
[-write_constant_for_signal_connection]
[-write_supply0_supply1]
```

Writes a netlist file of the design. By default, a logical netlist is saved. The netlist is in hierarchical Verilog format only if the original netlist loaded was hierarchical. The Innovus software saves the netlist from the top level of the hierarchy to the bottom level of the hierarchy if the original netlist is specified from top to bottom. If you run this command after using `loadDefFile`, it bundles bus pins to create busses in Verilog. Saving the netlist is done after running placement with timing driven option, clock tree synthesis, scan group re-order, or IPO.

You can use this command after importing the design.

Parameters

`-help`	Prints a brief description that includes the type and default information for each `write_netlist` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man write_netlist`.
`-abstract`	Writes an empty Verilog module with port declarations for the specified module. You can the `-abstract` parameter with the following parameters: • `-include_pg` • `-exclude_top_pg_ports` • `-derive_pg_port_dir` • `-export_top_pg_nets` • `-module` • `-phys` *Default*: Top module
`-apply_maskshift`	Adds a suffix to the cell name of a leaf instance based on the color assignments of the instance. *Data_type*: bool, optional **Note**: • The netlist must be unique or `-flat` must specified for the option to take effect. • The mask shifted netlist cannot be read back into Innovus, as the cell names with suffix do not actually exist in LEF and timing libraries.
`-cell_instance_color {true | false}`	Specifies to use the instance-based color. *Data_type*: enum, optional
`-cell_master_color {true | false}`	Specifies to use the cell master-based color from the Innovus color engine. *Data_type*: enum, optional

`-derive_pg_port_dir`	Specifies whether to derive (vs. assume) directions of implicit PG ports in saving a physical netlist. *Data_type*: bool, optional
`-exclude_cell_def` *cellName*	Excludes the module definition of the specified cells. *Data_type*: string, optional
`-exclude_ilm`	Excludes the ILM module.
`-exclude_insts_of_cells` *cellName*	Excludes the instances of the specified cells from the netlist. You can exclude logical or physical instances. The *cellName* can be a single cell name or a list of cell names separated by a space and surrounded by braces (`{}`) or double quotation marks (`""`). For example: `-exclude_insts_of_cells {cell1 cell2}` or `"cell1 cell2"`. *Data_type*: string, optional
`-exclude_leaf_cells`	Writes I/O instances, macro or block instances, and standard cell instances to the netlist, but does not include leaf cell definitions in the netlist. *Data_type*: bool, optional
`-exclude_top_pg_ports` *portName*	Excludes specified power and ground port(s) from the top cell's module port list. The *portName* can be a single port name or a list of port names separated by a space and surrounded by braces (`{}`) or double quotation marks (`""`). For example: `-exclude_top_pg_ports {VSS VDD}` or `"VSS VDD"`. *Data_type*: string, optional
`-export_top_pg_nets`	Specifies to use top-cell PG nets as ports for the top module in a physical netlist created with the `-phys` or `-include_pg` option. *Data_type*: bool, optional
file_name	Specifies the name of the file where the netlist is written. *Data_type*: string, required

`-flat`	Writes a flattened Verilog netlist. *Data_type*: bool, optional *Default:* If you omit this parameter, and if the original netlist loaded was hierarchical, the netlist is written in hierarchical Verilog format.
`-flatten_bus`	Writes a flattened-bus Verilog netlist. When this parameter is used, a bus bit, such as "A[0]" is output as "\A[0]". If such conversions conflict with existing object names, `write_netlist` warns about such conflicts and returns an error: **WARN: (ENCDB-2156): Cannot rename bus-bit %s '%s' in cell '%s' because the name '%s' is already in use on %s in that cell. Change the design to remove the conflicting name or run the command without the `-flatten_bus` option. **ERROR: (ENCSYC-1279): Cannot output flatten-bus netlist due to the reported object-name conflicts. *Data_type*: bool, optional
`-follow_logical_module_order`	Follows the logical module order. The order of the modules, written without `-include_pg`, in a hierarchical logical netlist is different from that, written with the option, in a hierarchical PG netlist. If `-follow_logical_module_order` is specified together with `-include_pg` (but not `-flat`) the order of the modules in a physical netlist will be the same as that in a logical netlist. **Note**: The same order is for non-leaf modules only.
`-ignore_fixed_mask`	Ignores cell fixedMask when maskShift is applied. *Data_type*: bool, optional
`-ilm`	Writes out a netlist with ILM guts. *Data_type*: bool, optional **Note:** This parameter can be specified only in the `flattenIlm` state. If this parameter is specified in the unflatten state, the software returns an error message.

`-include_bump_cells`	Writes bump cell information to the netlist. *Data_type*: bool, optional **Note:** This parameter is effective only when used with either the `-phys` or `-flat` parameters.
`-include_phys_cells` *cellName*	Includes the physical instances of the specified cells in the netlist. With this parameter, 1'b1/1'b0-style, logical connections are used for tie-off pins. The *cellName* can be a single cell name or a list of cell names separated by a space and surrounded by braces (`{}`) or double quotes (`""`), for example, `-include_phys_cells {cell1 cell2}` or `"cell1 cell2"`. *Data_type*: string, optional *Default*: If you omit this parameter, no physical cell module definitions will be in the netlist file. **Note:** `-include_phys_cells` creates a netlist that should not be read back into Innovus.
`-include_pg`	By default, PG connections in the original logical Verilog netlist are already written out (unless -flat is given). You can add "physical only" global PG net connections and "physical only" PG nets added by IEEE1801 to the Verilog output with this option. It adds PG nets inside each module and PG ports on each module in order to thread the PG connections to each pin. Some physical tools like LVS prefer to have these explicitly in the hierarchical Verilog. *Data_type*: bool, optional
`-include_phys_insts`	Activates physical netlisting by including physical instances, such as fillers, in the netlist. With this parameter, GNC-based/physical connections, such as vss, are used for tie-off pins. *Data_type*: bool, optional
`-keep_all_backslash`	Keeps all backslashes in names.

`-line_length` *`characters_per_line`*	Specifies the maximum number of characters per line for module port lists and instance port connections. When a line length exceeds the specified value, the software wraps the line at the next opportunity. You must specify a non-negative integer for the value. *`Default:`* `0` (The software wraps lines at the earliest opportunity.) *Data_type*: bool, optional
`-maskshift_suffix` *`maskShiftSuffix`*	Adds a user suffix to the cell name of a leaf instance instead of the tool's default suffix. *Data_type*: string, optional The default suffix is `_<layer>_<mask>`. For example, BUFX1_M0_1 default suffix means the M0 color flipped from mask 0 to mask 1. The `-mask_shift_suffix` option works only when `-apply_mask_shift` is used. **Note**: The netlist must be unique or `-flat` must specified for the option to take effect.
`-module` *`moduleName`*	Saves the specified module of the netlist. *Data_type*: string, optional
`-omit_floating_ports`	Omits the unconnected instance ports from the saved netlist. *Data_type*: bool, optional
`-only_leaf_cells`	Writes only leaf cell definitions in the netlist. *Data_type*: bool, optional *Default:* If you do not specify this parameter, the software writes leaf cell definitions, standard cell instances, and macro/block instances in the netlist file.
`-only_blocks`	Writes only macro or block cell definitions to the netlist. *Data_type*: bool, optional *Default:* If you do not specify this parameter, the software writes leaf cell definitions, standard cell instances, and macro/block instances in the netlist file.

`-only_std_cells`	Writes only standard cell instances in the netlist. *Data_type*: bool, optional *Default:* If you do not specify this parameter, the software writes leaf cell definitions, standard cell instances, and macro/block instances in the netlist file.
`-phys`	Writes out physical cell instances, and inserts power and ground nets in the netlist. This is used for LVS and for designs with multiple supply voltages. Hierarchical power and ground nets are also supported. *Data_type*: bool, optional **Note:** `-phys` creates a netlist that should not be read back into Innovus.
`-remove_power_ground`	Removes PG ports, PG nets, PG pin connections, and replaces PG net connections on signal pins with 1'b1 or 1'b0. *Data_type*: bool, optional
`-replace_inout_with_output_for_pg_ports`	Changes the direction of inout P/G ports to output except for the ones defined by IEEE1801/UPF. Their directions remain unchanged. *Data_type*: bool, optional
`-replace_wor_wand_with_wire`	Replaces wor/wand with wire statement. *Data_type*: bool, optional

`-top_module` *cellName*	Saves a module and modules instantiated in the hierarchy rooted at the specified module. Without `-top_module`, all modules, instantiated or not, are written for a logical netlist but only instantiated modules are written for a PG netlist:
	`write_netlist logical.v`
	`write_netlist -include_pg pg.v`
	Data_type: string, optional

`-top_module_first`	Writes out top module first, followed by the rest of the modules. *Data_type*: bool, optional **Note**: This feature is not available for physical netlist writing. As a result, the `-top_module_first` option will be ignored if any of the following parameters is also specified: `-phys`, `-include_pg`, and `-include_phys_insts`.
`-update_tie_connections`	Replaces `1'b1` and `1'b0` constants with undriven nets in the output netlist. Use this parameter if you may want to save a Verilog netlist from Innovus without any `1'b*` constants. *Data_type*: bool, optional **Note:** By default, the `write_netlist` command writes a Verilog netlist file of the design using `1'b*` constants to represent Assign nets and ports. However, subsequent verification steps that you use outside Innovus might not be able to handle these `1'b*` constants. In such a case, use `-update_tie_connections`.
`-use_cds_thru_for_assign`	Replaces the assign statement with cds_thru, the output netlist will not be able to read back by Innovus. This is only for third party tool. *Data_type*: bool, optional
`-use_eeq_cell_map {`*masterCell EEQcell1 EEQcell2*`}`	Defines the master cell and the EEQ cells that need to be mapped to the master cell {{masterCell EEQcell1 EEQcell2} ...} *Data_type*: string, optional

`-use_eeq_cell_with_liberty_info`	Specifies to use EEQ cell with .lib info as the master cell of a leaf instance. *Data_type*: bool, optional
`-write_constant_for_signal_connection`	Writes constant value for tie pin connection. When this option is specified, tool can assign the tie net name to related signal pin. If this option is not specified, tool assigns the real net name to the signal pin.
`-write_supply0_supply1`	When `-write_supply0_supply1` is on, PG nets are written as "supply0" or "supply1" when a net declaration is needed. `-write_supply0_supply1` is off by default. When it is off, a PG net is written as "supply0/supply1" only if it defined in the original Verilog. *Data_type*: bool, optional The `-write_supply0_supply1` option must be specified together with `-phys` or `-include_pg` to take the effect.

Example

- The following command writes out the netlist in Verilog format of the current state of the design in Innovus to a file `mydesign_cts.v`:

  ```
  write_netlist mydesign_cts.v
  ```

- The following command saves the netlist with `dtmf_recvr_core` as the top module to the `output.v` file:

  ```
  write_netlist -top_module dtmf_recvr_core output.v
  ```

- The following command saves the physical netlist of ILM internals to the `output.v` file:

  ```
  write_netlist -ilm -phys output.v
  ```

  Use this combination of options to pass on power or ground (P/G) information correctly to the netlist in low power ILM flow.

- The following example shows the netlist generated by `write_netlist -abstract`.

  ```
  module test(in1, in2, out1, out2, clk1, clk2, clk3, se1, se2);
   input [3:0] in1;
   input [7:0] in2;
   input clk1, clk2, clk3, se1, se2;
   output [3:0] out1;
   output [7:0] out2;
  endmodule
  ```

Related Information

- write_netlist Category Attributes
- Design Import and Export in Stylus chapter in the *Innovus Stylus Common UI User Guide.*
- Top-level Timing Closure Methodologies chapter in the *Innovus Stylus Common UI User Guide.*
  - Interactive Use of ILMs
- ECO Flows chapter in the *Innovus Stylus Common UI User Guide*
  - Pre-Mask ECO Changes from a New DEF File
  - Pre-Mask ECO Changes from an ECO File

write_netlist Category Attributes

The root attributes in the write_netlist category are:

- write_netlist_full_pin_out
- write_netlist_port_association_style

To set the value of any root attribute, use the `set_db` command:

```
set_db attribute_name value
```

To get the current value of any root attribute, use the `get_db` command:

```
get_db attribute_name
```

To get the current value of all attributes in the `write_netlist` category, use the following command:

```
get_db -category write_netlist
```

write_netlist_full_pin_out

`write_netlist_full_pin_out {true | false}`

Default: `false`

Data_type: bool, read/write

Specifies whether to write out every instance port connection, including unconnected ports unused in the input netlist.

write_netlist_port_association_style

`write_netlist_port_association_style {true | false}`

Default: `false`

Data_type: bool, read/write

Writes out the netlist with implicit port mapping (that is, ordered port connections or position-based port connections) for hierarchical instances.

write_stream

```
write_stream
stream_file_name
[-area {x1 y1 x2 y2} [-strict][-enclosed_instances_only]]
[-attach_inst_name attributeNumber]
[-attach_net_name attributeNumber]
[-attach_net_attribute {{prop_name1 attr_num1} {prop_name2 attr_num2........}}]
[-die_area_as_boundary]
[-format {stream oasis}[-output_inst_name instNAme] [-output_net_name netName]]
[-help]
[-ignore_subtype {NETTYPE SHAPE}]
[-lib_name libraryName]
[-map_file mapFile]
[-merge_via_cell_map_file cell_list]
[-merge {listOfExternalGDSOASISFiles} [-no_tsv tsv_name_list] [-uniquify_cell_names]]
[-report_file file_name]
[-mode {ALL | FILLONLY | NOFILL | NOINSTANCES}]
[-offset x y]
[-output_critical_net_color_from_file_only sideFile]
[-output_macros]
[-pvs_fill]
[-structure_name structureName | -no_structure_name]
[-unit {100 | 200 | 1000 | 2000 | 10000 | 20000}]
```

Creates a GDSII Stream file of the current database. Writes a summary of errors and warnings to the log file.

You can use the `write_stream` command after the design is placed, but is most commonly used only after the design is routed (which may also include metal fill shapes).

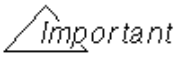
- If you do not specify a map file, the Innovus software creates a generic map file, in GDSII Version 3 format. The generic file shows the mapping that was used, if none was specified. You must customize the file to make it appropriate for your design. You can change the version by using the `write_stream_version` attribute.
- You must specify all layers to stream out. Layers that are not specified in the map file are not included in the file created by this command.
  Note: A cut layer in the map file (for a via) can have both MASK and SIZE at the same time.
- If you specify the `-merge` parameter and you want to merge files with a version number greater than `3`, the software generates a file with the highest version number from the merged files.
- If you do not specify `-output_macros`, `LEFPIN` and `LEFOBS` do not apply. If you do specify `-output_macros` and `LEFPIN` and `LEFOBS` are not specified in the map file for the layers in the LEF macros, the GDSII structures for those macros will be empty.

- If `MAXVOLTAGE` and `MINVOLTAGE` constructs are there in the layermap, `write_stream` will output a max voltage label and a min voltage label, for each net, to the GDSII file.

Parameters

`-area {x1 y1 x2 y2}`	Defines the area of the design to be output. Any shapes or instances enclosed, intersecting, or touching the specified area are output. So the data that is output can extend outside the specified area. By default the entire design is output. *Data_type*: rect, optional
`-attach_inst_name attributeNumber`	Specifies the instance name value added to the attribute on SREF of the specified attribute number. *Data_type*: int, optional *Range*: 1-127
`-attach_net_name attributeNumber`	Specifies that the net name will be added to the specified attribute number on the regular and special net wire shapes and vias SREF, but not on PIN shapes and pin vias. *Data_type*: int, optional *Range*: 1-127
`-attach_net_attribute {{prop_name1 attr_num1} {prop_name2 attr_num2........}}`	Specifies the attribute number on wiring shapes that will be used to store the value of the property associated nets in the design. For this option, you specify a list of property name and its attribute number. *Data_type*: string, optional **Note:** Multiple property names can be specified. Only wiring shapes associated with nets that contain one or more of the specified property names will have values assigned to their attributes. **Example:** `-attachPropName { {prop1 100} {prop2 101} }` In the above example, the attribute number `100` will be used to store the value of the property `prop1`, and the attribute number `101` will be used to store the value of the property `prop2`. It is a list of lists with a pair of property and its attribute number.

stream_file_name	Specifies the GDSII Stream output file. Include the `.gz` extension to enable file compression (for example, `GDS_file.gds.gz`). *Data_type*: list, required
`-die_area_as_boundary`	Forces the `write_stream` command to write the die area as a polygon (that is, a boundary). Otherwise, the `write_stream` command writes the die area as a rectangle (that is, a box). *Data_type*: bool, optional
`-enclosed_instances_only`	Specifies to stream out only those instances in which the shapes are fully enclosed by the specified area. *Data_type*: bool, optional
`-format {stream oasis}]`	Specifies the tape out file format. *Data_type*: enum, optional
`-help`	Outputs a brief description that includes type and default information for each `write_stream` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man write_stream`.

`-ignore_subtype {NETTYPE SHAPE}`	gnores the map file line containing the subtype keyword , and warning message is printed in the log file. *Data_type*: dict, optional For example, in the example below, with `-ignore_subtype {NETTYPE SHAPE}` the line 61, 62, and 63 of the map file will be ignored. • Line61: M2 SPNET:NETYPE:POWER:MASK:2 32 12 • Line62: M2 SPNET:SHAPE:STRIPE:MASK:2 32 22 • Line63: M2 SPNET: NETYPE:GROUND:SHAPE:STRIPE:MASK:2 32 32 • Line64: M2 SPNET:MASK:2 32 2
`-lib_name` *`libraryName`*	Specifies the library to convert to GDSII Stream format. *Default:* `DesignLib` *Data_type*: string, optional
`-map_file` *`mapFile`*	Specifies the file used for layer mapping. This file is required for successful stream out, and is read by the next tool used in the design flow.  If you do not specify a file with this parameter, the Innovus software will read a `write_stream.map` file if one exists. If a `write_stream.map`file does not exist, the software creates a generic map file that you must customize for your design. For information on the map file format, see "GDSII or OASIS Map File Format" in the Design Import and Export in Stylus chapter of the *Innovus Stylus Common UI User Guide*. *Data_type*: string, optional

`-merge_via_cell_map_file` *`cell_list`*	Specifies the list of cells that need merge via cell from an external file. *Data_type*: string, optional
`-merge` **{***`listOfExternalGDSOASISFiles`***}**	Specifies a single file or list of files to merge. Checks all merged files for name collisions, and generates a warning if any names are changed. Uses the following conventions for new names: • Replaces dot separators (`.`) with underscores (_). For example, `test.test_gds.gds` is renamed `test_test_gds_gds`. • Adds unique identifiers. For example, if more than one cell is named `test.test.gds.gds`, they are renamed `test_test_gds_gds_1`, `test_test_gds_gds_2`, and so on. *Data_type*: string, optional Supports the mixed format of the GDSII and OASIS files. You need to specify the merged files with mixed format using the `-merge` option. The tool will identify the files format, covert them, and merge them to a stream out file. Compressed files are acceptable (include the `.gz` extension to enable file compression). The Innovus software automatically creates blackboxes when merging and writing macro files. It ignores any cells in the merge files that are not used in the design. If you specify more than one file, separate the file names with spaces and enclose the list of file names in braces (`{}`) or double quotation marks (`"`) You can use a wildcard to read all `.gds` files in a specified directory. The usage is as follows: `write_stream -merge GDSIIdir/*.gds` where *`GDSIIdir`* is the name of the directory that contains the `.gds` files. ***Use caution when using wildcards, as they might affect the order in which the software reads merge files.***

`-mode {ALL | FILLONLY | NOFILL | NOINSTANCES}`	
	Identifies the layers to write. *Data_type*: enum, optional Specify one of the following: • `ALL`--Writes all layer information specified in mapFile as follows: ◦ All instances ◦ All via instances ◦ All generated via cells • `FILLONLY`--Writes only fill layers specified in mapFile as follows: ◦ No instances ◦ Only fill via instances ◦ Only generated fill via cells • `NOFILL`--Writes only non-fill layers specified in mapFile. • `NOINSTANCES`--Writes wiring (including vias) only; does not write out `COMPONENT` instances.
`-no_structure_name`	Enables you merge to files without loading an Innovus database when the `-merge` parameter is also specified. *Data_type*:
`-no_tsv` *`tsv_name_list`*	Specifies a list of TSV via names that should not be written out. *Data_type*: string, optional

`-offset` *x y*	Specifies, in microns, the location of the design in the GDSII output file. When the GDSII output file is saved, the design shifts to the specified location. *Data_type*: point, optional

`-output_critical_net_color_from_file_only` *sideFile*	Specifies the net list and the layer range through a side-file. The `write_stream` command outputs the object color on the specified layers and nets. Other objects remain gray in GDS.
	Here is an example of a side file:
	`<LAYERMAP>`
	`M5`
	`PIN:MASK:1,NET:MASK:1,SPNET:MASK:1,VIA:MASK:1`
	`35 271`
	`M5`
	`PIN:MASK:2,NET:MASK:2,SPNET:MASK:2,VIA:MASK:2`
	`35 272`
	`M6`
	`PIN:MASK:1,NET:MASK:1,SPNET:MASK:1,VIA:MASK:1`
	`36 271`
	`M6`
	`PIN:MASK:2,NET:MASK:2,SPNET:MASK:2,VIA:MASK:2`
	`36 272`
	`</LAYERMAP>`
	`<NETNAMELIST>`
	`TOP/NET1`
	`TOP/NET2`
	`...`
	`</NETNAMELIST>`

`-output_inst_name` *instNAme*	Writes the instance name to the oasis file. *Data_type*: string, optional

`-output_macros`	Writes LEF abstract information such as LEF pin geometries and obstructions for macros. Writes one text label per port within a LEF macro pin plus one text label per shape for feedthrough pins. Specify this parameter to create GDSII output that contains the LEF macro structures as well as the design data. You can also use this parameter to debug LEF macro definitions against verification DRC rules. The output contains macro pins and obstructions. **Note:** `LEFPIN` and `LEFOBS` apply only when you specify this parameter. If you specify this parameter and `LEFPIN` and `LEFOBS` are not specified in the map file for the layers in the LEF macros the GDSII structures for those macros will be empty. *Data_type*: bool, optional
`-output_net_name` *netName*	Writes the net name to the oasis file. *Data_type*: string, optional
`-pvs_fill`	Merges the PVS fill data attached to the current DB into the output GDS file and instantiates the PVS top fill cell at 0,0 R0 inside the top design cell. The PVS fill format must be GDS. *Data_type*: bool, optional
`-report_file file_name`	Outputs cell name changed to the file. *Data_type*: string, optional
`-strict`	Truncates the object outside the area. *Data_type*: bool, optional
`-structure_name` *structureName*	
	Specifies the superstructure that consists of all geometries that the map file writes out. *Default:* Current design's top cell name *Data_type*: string, optional

`-uniquify_cell_names`	Creates unique cell names if there are name collisions when the software merges files. Creates unique cell names if there are name collisions when the Innovus software merges files. *Data_type*: bool, optional ⓘ You must specify the `-merge` parameter if you specify this parameter. When you list the merge files, list the top-level file first, as the software uses the first name in the search path when renaming cells. When a cell name is repeated, the software renames the cell *`cellName_filename`*, where *`filename`* is the name of the merge file in which the cell name appears. The software changes the name of references to that cell in the merge file. The software changes the reference name if the referenced cell is defined in the same file, regardless of whether the cell is defined before or after the reference. If a reference in a merge file refers to a cell definition in a different merge file, then the reference is not changed in the GDSII Stream output file. *Default:* If you do not specify this parameter, the software ignores duplicate cell names. **Note:** You must specify the `-merge` parameter if you specify this parameter.
`-unit {100 | 200 | 1000 | 2000 | 10000 | 20000}`	Specifies the resolution for values in the GDSII file. Choose `100`, `200`, `1000`, `2000`, `10000`, or `20000`. *Default:* Units specified in LEF file *Data_type*: int, optional

Examples

- The following command creates a GDSII Stream file (merging cells from the ios.gds file):

```
write_stream amba_usb.gds -map_file write_stream.map -lib_name DesignLib
  -attach_inst_name 12 -merge {ios.gds}
  -attach_net_name 15
  -die_area_as_boundary
```

- The following command creates blackboxes for cells not found in the merge libraries and writes macros only for those cells:

  ```
  write_stream newFile.gds.gz -merge {myFile1 myFile2 myFile3} -output_macros
  ```

- The following command merges cells needed in the design from all files in `myDirectory` with the current database:

  ```
  write_stream myFile.gds.gz -merge myDirectory/*.gds
  ```

  If a merge file contains cells such as `A`, `B`, `C`, `D`, `X`, `Y`, and `Z`, and the design contains only cells `A`, `B`, and `D`, then the Innovus software merges only cells `A`, `B`, and `D`.

- The following example shows a stream out map file. The first column specifies the layer object name, and the second column specifies the layer object type. The third and fourth columns specify the GDS layer and data type to stream those objects, respectively. When the object name is "NAME", it indicates the text label for the object types specified in the second column.

  ```
  ==========original streamout.map file=============

  M1         NET          1         0

  M1         SPNET        2         0

  M1         PIN          3         0

  M1         LEFPIN       4         0

  M1         FILL         5         0

  M1         VIA          6         0

  M1         VIAFILL      7         0

  M1         LEFOBS       8         0

  NAME       M1/NET       1         0 <--- text for nets

  NAME       M1/SPNET     2         0 <--- text for special nets (i.e. power nets)

  NAME       M1/PIN       3         0 <--- text for pins including top level IO pins

  NAME       M1/LEFPIN    4         0 <--- text for LEF pins

  ================================================
  ```

Related Information

- write_stream Category Attributes
- Design Import and Export in Stylus chapter in the *Innovus Stylus Common UI User Guide*

write_stream Category Attributes

The root attributes in the write_stream category are:

- write_stream_allow_path_type4
- write_stream_cell_instance_color
- write_stream_cell_master_color
- write_stream_cell_name_prefix
- write_stream_cell_name_suffix
- write_stream_check_map_file
- write_stream_compatible
- write_stream_define_via_name
- write_stream_ignore_fixed_mask
- write_stream_label_all_pin_shape
- write_stream_oasis_compression
- write_stream_oasis_layer_name
- write_stream_oasis_cell_offset
- write_stream_merge_append
- write_stream_pin_text_orientation
- write_stream_remove_nets
- write_stream_snap_to_mfg
- write_stream_stream_convert_rect_to_path
- write_stream_stream_version
- write_stream_text_size
- write_stream_uniquify_cell_names_prefix
- write_stream_via_names
- write_stream_virtual_connection

To set the value of any root attribute, use the `set_db` command:

```
set_db attribute_name value
```

To get the current value of any root attribute, use the `get_db` command:

```
get_db attribute_name
```

To get the current value of all attributes in the `write_stream` category, use the following command:

```
get_db -category write_stream
```

write_stream_allow_path_type4

`write_stream_allow_path_type4 {true | false}`

Default: `true`

Data_type: bool, read/write

Converts `PATHTYPE4` to `PATHTYPE0`. If you specify `write_stream_allow_path_type4 0`, the software converts `PATHTYPE4` paths to`PATHTYPE0` when outputting the path.

write_stream_cell_instance_color

`write_stream_cell_instance_color {true | false}`

Default: `false`

Data_type: bool, read/write

Streams out the instance-based color. The instance mask shift is honored in `write_stream`. When the option is on and the `-merge` option is used, the merged gds files with mask shift suffix must be provided through the `-merge` option.

write_stream_cell_master_color

`write_stream_cell_master_color {true | false}`

Default: `true`

Data_type: bool, read/write

Streams out the cell-based color from the Innovus color engine.

write_stream_cell_name_prefix

`write_stream_cell_name_prefix` *`prefix`*

Default: `""`

Data_type: string, read/write

Specifies to add a prefix to ALL cell names being written out, use user-specified prefix can avoid potential conflict when integrating different macros, which potentially two different cells (in two different libraries) with the same name.

write_stream_cell_name_suffix

`write_stream_cell_name_suffix` *`suffix`*

Default: `""`

Data_type: string, read/write

Specifies to add a prefix to ALL cell names being written out, use user-specified prefix can avoid potential conflict when integrating different macros, which potentially two different cells (in two different libraries) with the same name.

Specifies to add user specified suffix instead of appending color pattern to normal cell names. This applies to fixed color shift (flip) pattern only. This means that only single shift (flip) pattern is applied to the design.

write_stream_check_map_file

`write_stream_check_map_file {true|false}`

Default: `false`

Data_type: bool, read/write

Checks the specified map file. If the objects exist in Innovus and but therse objects are not mapped in the map file, the missed objects mapping and missed objects are printed in the log file. Tool checks only the main objects mapping.

write_stream_compatible

`write_stream_compatible {true | false}`

Default: `false`

Data_type: bool, read/write

Output instance only if `FOREIGN` is specified.

write_stream_define_via_name

`write_stream_define_via_name {default |` *`format_string`*`}`

Default: `%t_VIA%u` (This default value is used so that the names of the via cells are compatible with versions 4.2 and earlier versions of the Innovus software.)

Data_type: string, read/write

Specifies the naming convention that the `write_stream` command uses for via cells declared in the DEF file.

Note: When the version of the GDS file is 3 (this is the default version), via names are truncated if they are longer than 32 characters. The software automatically appends a unique identifier to names if there is a name conflict.

You can use any of the following characters to specify a format string to generate names that give you information about the cell contents. Separate the characters with underscores (_) to improve readability. You can list them in any order.

- `%c`
  Specifies the number of columns.
- `%l(lcu)`
  Specifies the via layers. For the lower layer, use `%l` or `%l(l)`. For the cut layer, use `%l(c)`. For the upper layer, use`%l(u)`. You can specify one, two, or all three layers. You must enclose the `l`, `c`, and `u` in parentheses, as in the following examples: `%l(c)`, `%l(lu)`.
- `%n`
  Specifies the number of cut rows and columns that make up a cut array.
- `%r`
  Specifies the number of cut rows.
- `%t`
  Specifies the top structure name.
- `%u`
  Specifies a unique number for the via.
- `%v`
  Specifies the via name in the DEF file.

For more information, see "Vias," in the DEF Syntax chapter of the *LEF/DEF Language Reference*.

write_stream_ignore_fixed_mask

```
write_stream_ignore_fixed_mask {true | false}
```

Default: `false`

Data_type: bool, read/write

Ignore macro FIXEDMASK keywords, enable color flip function.

write_stream_label_all_pin_shape

```
write_stream_label_all_pin_shape {true | false}
```

Default: `false`

Data_type: bool, read/write

Labels all pin shapes, whether or not they are physically connected, when set to `true`.

write_stream_oasis_compression

```
write_stream_oasis_compression {true | false}
```

Default: `true`

Data_type: bool, read/write

Reduces file size by using repetition compression. Repetition compression supports an 'array' of cell placements, geometries, or text elements, instead of write each the element separate only different in position, this can save duplicate information for an OASIS layout file size.

write_stream_oasis_layer_name

```
write_stream_oasis_layer_name {true | false}
```

Default: `false`

Data_type: bool, read/write

Support LAYERNAME in OASIS file.

write_stream_oasis_cell_offset

```
write_stream_oasis_cell_offset {true | false}
```

Default: `false`

Data_type: bool, read/write

Includes cell offset table to declare the byte offset from the beginning of the file (byte 0) to the corresponding CELL record location that appears in the file.

write_stream_merge_append

```
write_stream_merge_append {true | false}
```

Default: `false`

Data_type: bool, read/write

Append cell color information in current design to cell geometries that are from merging files. This option is for the fixed-mask design.

write_stream_pin_text_orientation

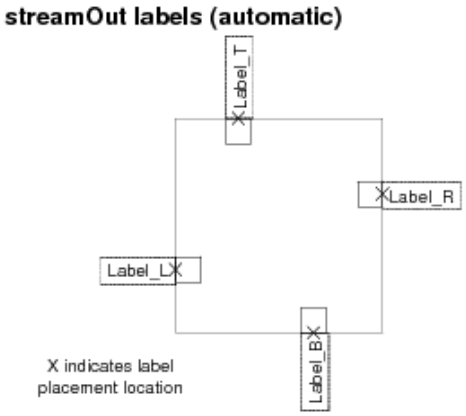
```
write_stream_pin_text_orientation {default | automatic}
```

Default: default

Data_type: string, read/write

Controls the pin text orientation, making it more readable.

- `default` implies that the labels are centered in the pin shapes and have `R0` orientation and `LowerLeft` justification.
- `automatic` implies that the labels for pin shapes that are abutted to the design boundary are centered on the edge of the pin shape.



write_stream_remove_nets

`write_stream_remove_nets {`*`list_of_nets`*`}`

Default: `""`

Data_type: string, read/write

Removes the specified nets from the output file.

write_stream_snap_to_mfg

`write_stream_snap_to_mfg {true | false}`

Default: `false`

Data_type: bool, read/write

Specifies whether to snap the nets' wires to the manufacturing grid.

write_stream_stream_convert_rect_to_path

`write_stream_stream_convert_rect_to_path {true | false}`

Default: `false`

Data_type: bool, read/write

Converts DEF SPECIALNETS path/rectangle and FILLS shapes to GDSII/stream PATH data instead of BOUNDARY data when enabled. Conversion to path data can result in significantly smaller output database when there are many rectangle shapes (example: after metal fill is included).

write_stream_stream_version

```
write_stream_stream_version version
```

Default: `3`

Data_type: int, read/write

Specifies the version for a created GDSII Stream file. If you use the `write_stream -merge` parameter, `write_stream` uses the highest version of the merged files, instead of the value specified by the parameter.

write_stream_text_size

```
write_stream_text_size value
```

Default: `1`

Data_type: double, read/write

Changes the size of the text used in the text labels in the GDSII Stream file written by the Innovus software.

- To increase the size of the text, specify a positive real number that is greater than 1, as in the following example:
  ```
  write_stream_text_size 10.0
  ```
- To decrease the size of the text, specify a positive real number that is less than 1, as in the following example:
  ```
  write_stream_text_size 0.5
  ```

Note: The write_stream_text_size attribute affects the size of only automatically created text. It does not affect text created through `create_custom_text`.

write_stream_uniquify_cell_names_prefix

```
write_stream_uniquify_cell_names_prefix {true | false}
```

Default: `false`

Data_type: bool, read/write

Adds a prefix, instead of a suffix, to uniquified cell names. The prefix is the source filename. If this parameter is not specified, the software adds a suffix (also the source filename) to uniquified cell names.

write_stream_via_names

```
write_stream_via_names {true | false}
```

Default: `false`

Data_type: bool, read/write

Determines the naming convention used for vias declared in the LEF file.

- Specify `true` to create unique names.
- Specify `false` to retain LEF via names.

write_stream_virtual_connection

```
write_stream_virtual_connection {true | false}
```

Default: `true`

Data_type: bool, read/write

Specifies whether a colon (`:`) label is appended to pin names for DEF pins with the `.extra`*N* syntax and to LEF pins with multiple ports (if `-output_macros` is specified for LEF pin ports that have disjoint shapes).
Default: `true`

- Specify `true` to append a colon to pin names.
- Specify `false` not to append a colon to pin names.

Note: For more information, see the DEF Syntax chapter of the *LEF/DEF Language Reference*.

write_via_defs

```
write_via_defs
[-help]
[-generate]
[out_file filename]
[-all_vias | -signal | -all_rules | -default | -ndr ]
```

Helps in debugging and testing. All the boolean options are default to false. If none is set, the vias are dumped out in LEF format.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `write_via_defs` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man write_via_defs`.
`-all_rules`	Dumps out vias from all routing rules. *Data_type*: bool, optional
`-all_vias`	Dumps out all vias. *Data_type*: bool, optional
`-generate`	Dumps out auto-generated vias. *Data_type*: bool, optional
`-default`	Dumps out vias of default routing rule.
`out_file filename`	Specifies the output file name. *Default*: `dump_via.out` *Data_type*: string, optional
`-ndr`	Dumps out vias of nondefault routing rules. *Data_type*: bool, optional
`-signal`	Dumps out signal vias. *Data_type*: bool, optional

Example

The following command dumps out all signal vias to the default output file, `dump_via.out`:

```
write_via_defs -signal
```

Here's the sample output:

```
#### Begin signal vias ####
VIA V01  # LEF
  LAYER POLY ; RECT -0.033 -0.033 0.033 0.033 ;
  LAYER M1L ; RECT -0.033 -0.033 0.033 0.033 ;
  LAYER C01L ;
    RECT -0.033 -0.033 0.033 0.033 ;
END V01
VIA V01N  # LEF
  LAYER NDIFF ; RECT -0.033 -0.033 0.033 0.033 ;
  LAYER M1L ; RECT -0.033 -0.033 0.033 0.033 ;
  LAYER C01L ;
    RECT -0.033 -0.033 0.033 0.033 ;
END V01N
VIA V01P  # LEF
  LAYER PDIFF ; RECT -0.033 -0.033 0.033 0.033 ;
  LAYER M1L ; RECT -0.033 -0.033 0.033 0.033 ;
  LAYER C01L ;
    RECT -0.033 -0.033 0.033 0.033 ;
END V01P
VIA V1L2L  # LEF
  LAYER M1L ; RECT -0.033 -0.033 0.033 0.033 ;
  LAYER M2L ; RECT -0.033 -0.033 0.033 0.033 ;
  LAYER C1L2L ;
    RECT -0.033 -0.033 0.033 0.033 ;
END V1L2L
VIA V2L3L  # LEF
  LAYER M2L ; RECT -0.033 -0.033 0.033 0.033 ;
  LAYER M3L ; RECT -0.033 -0.033 0.033 0.033 ;
  LAYER C2L3L ;
    RECT -0.033 -0.033 0.033 0.033 ;
END V2L3L
VIA V3L4L  # LEF
  LAYER M3L ; RECT -0.033 -0.033 0.033 0.033 ;
  LAYER M4L ; RECT -0.033 -0.033 0.033 0.033 ;
  LAYER C3L4L ;
    RECT -0.033 -0.033 0.033 0.033 ;
.......................
.......................
.......................
END SP4V4L5LHVR1S5
VIA SP4V4L5LHVR3S5  # LEF
  LAYER M4L ; RECT -0.066 -0.033 0.264 0.033 ;
  LAYER M5L ; RECT -0.033 -0.066 0.033 0.066 ;
  LAYER C4L5L ;
    RECT -0.033 -0.033 0.033 0.033 ;
END SP4V4L5LHVR3S5
#### End signal vias ####
```

report_gates

```
report_gates
[-help]
[-hinst hInstName]
[-out_file fileName]
[-sort {name count area leakage_power internal_power total_power switching_power}] [-power | -
leakage_power ]
```

Reports the technology library cells that were implemented (and identifies their originating libraries), the area of the cell instances, and the break up of the instances into timing models, sequential cells, integrated clock-gating cells, inverters, buffers, and logic gate cells. Optionally power information can be reported.

The result is based on the cell_area calculated from LIB information. Use set_db use_area_from_lef true (default: false), if you want to get correlation between reports generated with `report_gates` command and `report_qor` or `report_utilization` commands. When a lef is read into Genus, the property "use_area_from_lef" is automatically set to `true`.

Timing models can refer to memory cells, IPs, and so on.

Parameters

`-help`	Prints a brief description that includes the type and default information for each `report_gates` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man report_gates`
`-hinst hInstName`	Specifies the name of the hinst to collect the cell information. Only cells in the specified hinst will be considered.
`-leakage_power`	Provides leakage power statistic only.
`-out_file fileName`	Specifies the file in which report will be saved. If not specified, information is dumped in the log.
`-power`	Provides internal, switching, leakage and total power statistic.
`-sort {name count area leakage_power internal_power total_power switching_power}`	Specifies the sorting criteria to sort content of the tables to generate leakage_power/internal_power value only respected if the -power option is also specified.

Examples

```
>report_gates -sort name -out_file check.rpt
```

```
#####################################################################
#  Generated by:      Cadence Innovus 22.10-a047_1
#  OS:                Linux x86_64(Host ID sjcvl-swatip)
#  Generated on:      Mon Jun 20 09:53:59 2022
#  Design:            carve
#  Command:           report_gates -sort name -out_file check.rpt
#####################################################################
Library cell information:
 ------------------------------------------
| Cell          |  Instance  |  Area(um^2)  |
|------------------------------------------|
| LDL_AO22_3    |          1 |        1.882 |
| LDL_AO2B2_S_1 |          5 |        9.409 |
| LDN_BUF_2     |          1 |        0.941 |
| LDN_BUF_HD_3  |          2 |        2.195 |
|------------------------------------------|
| Total         |          9 |       14.427 |
 ------------------------------------------
Library cell type information:
 ------------------------------------------------------
| Cell Type        |   Instance   |        Area         |
|                  |              |       um^2   |   %  |
|------------------------------------------------------|
| Buffer           |            0 |       0.000 |   0.0 |
| Clock Gater      |            0 |       0.000 |   0.0 |
| Combinational    |            0 |       0.000 |   0.0 |
| Flop             |            0 |       0.000 |   0.0 |
| Inverter         |            0 |       0.000 |   0.0 |
| Latch            |            0 |       0.000 |   0.0 |
| Macro            |            0 |       0.000 |   0.0 |
| Multi-Bit Flop   |            0 |       0.000 |   0.0 |
| Physical Cell    |            9 |      14.427 | 100.0 |
|------------------------------------------------------|
| Total            |            9 |      14.427 | 100.0 |
 ------------------------------------------------------
```

report_sequential

```
report_sequential
[-help]
[-deleted]
[-full]
[-hinst hInst_names]
[-mapped]
[-optimized]
[-out_file <string>]
```

Reports information about sequentials the design. Sequentials are the elements that have a library is_sequential tag. These can be flops, latches, and clock gates.

Parameters

`-help`	Prints a brief description that includes the type and default information for each `report_sequential` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man report_sequential`
`-deleted`	Generates a report with the deleted flops and the reason.
`-full`	Generates a report with the sequential name, whether it was inferred or instantiated, the type of sequential, and the type of reset.
`-hinst` *`hInst_names`*	Hinst for the report. By default, the design is reported.
`-mapped`	Generates a report with synthesis requirements modified to map the sequential.
`-optimized`	Generates a report with sequentials optimized to constants or are unloaded, but have not been removed.
`-out_file` *`file_name`*	File to dump the report. Default is stdout

Example

- The following command reports the deleted flops:

```
innovus 1> report_sequential -deleted
Sequential elements: deleted flops
-------------------------------------

  Reason      Instance Name

-------------------------------------

unloaded       foo/bar/q_reg[15]
```

```
unloaded       foo/bar/q_reg[31]
unloaded       foo/bar/q_reg[47]
unloaded       foo/bar/q_reg[63]
unloaded       foo/bar/q_reg[79]
constant 0     moo/ff_reg_9
```

- The following command reports the optimized flops:

```
innovus 2> report_sequential -optimized
Sequential elements: optimized flops
----------------------------------------

  Reason      Instance Name

----------------------------------------
constant 0     RESULTS_CONV_INST/out_p1_reg[7]
constant 0     TDSP_CORE_INST/EXECUTE_INST/alu_cmd_reg[3]
constant 0     TDSP_CORE_INST/EXECUTE_INST/read_data_reg
constant 0     TDSP_CORE_INST/EXECUTE_INST/read_port_reg
```

check_design Category Attributes

The root attribute in the `check_design` category is:

- check_design_across_hierarchy
- check_design_report_async_pins
- check_design_report_scan_pins
- check_design_threshold_fanout

To set the value of any root attribute, use the `set_db` command:

```
set_db attribute_name value
```

To get the current value of any root attribute, use the `get_db` command:

```
get_db attribute_name
```

To get the current value of all attributes in the `read_physical` category, use the following command:

```
get_db -category check_design
```

check_design_across_hierarchy

`check_design_across_hierarchy {true | false}`

Default: `true`

Data_type: boolean, read/write

Checks for undriven/multi-driven/unloaded/constant pins/ports by checking across hierarchical boundaries. By default checks are across hierarchy.

check_design_report_async_pins

`check_design_report_async_pins {true | false}`

Default: `true`

Data_type: boolean, read/write

Checks async pins in a netlist design.

check_design_report_scan_pins

`check_design_report_scan_pins {true | false}`

Default: `true`

Data_type: boolean, read/write

Checks scan pins in a netlist design.

check_design_threshold_fanout

check_design_threshold_fanout `{true | false}`

Default: `true`

Data_type: boolean, read/write

Checks for upper threshold in a netlist design.

17

Interactive ECO Commands and Attributes

- add_io_buffers
- add_notch_fill
- connect
- connect_hpin
- connect_pin
- create_diode
- create_hinst
- create_hnet
- create_hport
- create_inst
- create_module
- create_net
- create_port
- delete_assigns
- delete_bus_net
- delete_empty_hinsts
- delete_feedthru_buffer
- delete_hinst
- delete_hpin
- delete_inst
- delete_nets
- delete_notch_fill
- disconnect
- disconnect_hpin
- disconnect_pin
- eco_add_repeater

- eco_compare_netlist
- eco_delete_repeater
- eco_design
- eco_oa_design
- eco_read_def
- eco_remove_tied_inputs
- eco_swap_spare_cell
- eco_update_cell
- eco Category Attributes
- ECO Without Category Attributes
- end_eco
- fix_multi_drivers
- group
- gui_show_buffer_tree
- init_eco
- place_eco
- read_eco
- ungroup
- update_inst
- bitblast_ports
- delete_unintantiated_modules

add_io_buffers

```
add_io_buffers
[-help]
[-base_name baseName]
[-dishonor_power_intent_domain_boundary_check_for_iso_level_shifter_strategy]
[-exclude_clock_nets]
{[-in_cells cellNameList][-out_cells cellNameList]}
[-include_nets_file selNetFileName]
[-exclude_nets_file excNetFileName]
[-port]
[-pre_place]
[-suffix suffixName]
[-skip_refine_place]
[-status {placed fixed softfixed}]
```

Adds buffers/inverters to the I/O pins of a block and places the buffers/inverters near the I/O pins. When a top-level input (output) is connected to a power domain other than the default, the added buffer is matched to the hierarchy of the destination (source) cell and placed inside the power domain.

This command is power-domain aware. To take advantage of this feature when the netlist contains level shifters, load the shifter table before running this command.

Through the `-port` and `-suffix` parameters, you can control the instance and net names created when you use `add_io_buffers`.

You can use this command after loading a design.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `add_io_buffers` parameter. For a detailed description of the command and all of its parameters, use the man command: `man add_io_buffers`

`-base_name` *baseName*	Specifies the base name of the new nets. For nets with the same base name, you should consider the hierarchical index and add `1`. For example, if there are several nets with an `abc` base name, ending on `abc98`, then the first new net with the same base name is `abc99`. *Data_type*: string, optional
`-dishonor_power_intent_domain_boundary_check_for_iso_level_shifter_strategy`	Dishonors power-domain boundary check for isolation/level-shifter instances. *Data_type*: bool, optional
`-exclude_clock_nets`	Does not connect the buffer on clock nets. You must trigger the clock net markings prior to running `add_io_buffers`, with the `time_design` command. *Data_type*: bool, optional
`-exclude_nets_file` *excNetFileName*	Specifies the file that contains the names of nets to exclude from the buffer attachment operation. *Data_type*: string, optional

`-in_cells` *cellNameList*	Specifies the cell name or a list of cell names of the input (`-in_cells`) buffer. If the specified cell is an inverter, then the `add_io_buffers` command inserts an inverter pair. *Data_type*: string, optional **Note:** If the `-in` and `-out` parameters are specified together, both input and output pins will be attached buffers/inverters. If only `-in` (or `-out`) option is specified, then only input (or output) pins will be attached buffer. If a list of cells is specified, it will insert the first CPF compliant cell. It is recommended to give non-AO buffers/inverters before AO buffers/inverters in the list.

`-out_cells` *cellNameList*	Specifies the cell name or a list of cell names of the output (`-out_cells`) buffer. If the specified cell is an inverter, then the `add_io_buffers` command inserts an inverter pair. *Data_type*: string, optional **Note:** If the `-in` and `-out` parameters are specified together, both input and output pins will be attached buffers/inverters. If only `-in` (or `-out`) option is specified, then only input (or output) pins will be attached buffer. If a list of cells is specified, it will insert the first CPF compliant cell. It is recommended to give non-AO buffers/inverters before AO buffers/inverters in the list.
`-port`	Prepends the port name to the name of the net or instance created. *Data_type*: bool, optional

`-pre_place`	Specifies to legally place the inverter connected to the port and to leave the other inverter unplaced. *Data_type*: bool, optional
`-include_nets_file` *selNetFileName*	Specifies the file that contains the names of nets to include in the buffer attachment operation. *Data_type*: string, optional
`-skip_refine_place`	Skips legalization of the new buffer/inverter (incremental placement needs to be run later to legalize it). *Data_type*: bool, optional
`-status {placed fixed softfixed}`	Specifies the placement status of the added buffer as placed/fixed/softfixed. *Data_type*: enum, optional
`-suffix` *suffixName*	Appends a string to name of the net or instance created. *Data_type*: string, optional

Example

- The following command attaches buffers `bufx4` to the I/O pins. The new nets have a basename of `abc`:

  ```
  add_io_buffers -base_name abc -in_cells bufx4
  ```

- The following command prepends the port name and appends the `_buf` string to the name of the instance and net created by `add_io_buffers`:

  ```
  add_io_buffers -port -suffix "buf"
  ```

  Resulting instance name:

  ```
  port1_I_buf
  ```

  Resulting net name:

  ```
  port1_N_buf
  ```

Related Information

- `delete_buffer_trees`
- `time_design`

add_notch_fill

```
add_notch_fill
[-help]
[-area {x1 y1 x2 y2}]
[-report filename]
[-use_existing_drc_result]
[-use_non_default_spacing]
```

Detects and fills notches, gaps, and holes, and corrects violations for rectangle notches. Also, the support is limited nodes 28nm and below.

Creates geometries in `DRCFILL` shapes as special wires attached to each net. Saves the geometries in DEF or GDSII Stream format. In a DEF file, the geometries are saved in the special net section. The software considers the geometries as active design objects.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `add_notch_fill` parameter. For a detailed description of the command and all of its parameters, use the man command: `man add_notch_fill`
`-area {x1 y1 x2 y2}`	Specifies the area to check and fill. For example, `add_notch_fill -area {0 0 100 100}`. *Data_type:* rect, optional
`-report filename`	Specifies the report file. The file includes the locations, types and number of violations corrected. For example, `add_notch_fill -report design_fill.rpt`. *Default:* `design.fill.rpt` *Data_type:* string, optional
`-use_existing_drc_result`	Allows fixing the `convexCornerSpacing` issues based on Verify Geometry markers. *Data_type:* bool, optional
`-use_non_default_spacing`	Honors nondefault spacing rules when filling notches. This behavior matches the behavior of `check_drc`; that is, when this parameter is specified, `add_notch_fill` honors both default and nondefault spacing rules. If this parameter is not specified, `add_notch_fill` honors default spacing rules but does not honor nondefault spacing rules. *Data_type:* bool, optional

Example

- The following commands check the notch issues and create drc markers with verify_drc, and add drc fill shapes on the drc marker location:

```
check_drc
add_notch_fill -use_existing_drc_result
```

Related Information

- `check_drc`
- `delete_notch_fill`
- ECO Flows chapter in the *Innovus Stylus Common UI User Guide*

connect

```
connect
[-help]
port|pin|hpin|hport|pg_pin+
[-net net_object | -net_name net_name | -constant | -hnet hnet_object  ]
```

Connects objects together. If these objects exist in different levels of hierarchies, hpins/hport will be created to make these connections.

If more than one object has a driver (that is has a net connected), it creates a multi driver situation. If only one object has a net, all other objects will be connected to that net. If no object has a net, a net will be created and they all will be connected to that net and the `-net_name` option provides a mechanism to optionally name the new net. Otherwise, a new net will be created using n_<i> convention. If the `-net` option is used (along with a net object), then all objects will be disconnected from their existing connection and connected to the new net. For local connections, it returns then hnet object pointer used to make the connection. For hierarchical connection, it returns the parent net object pointer.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `connect` parameter. For a detailed description of the command and all of its parameters, use the man command: `man connect`
port|pin|hpin|hport|pg_pin+	List of objects to connect. Multiple pin/port objects are supported. Note that only object pointers like hport:h1/hport1 are allowed, and simple names like h1/hport1 are not allowed. All objects in the list will be connected together so if objects have existing drivers, then a multidriver situation can be created. *Data_type:* port| pin| hpin| hport| pg_pin)+, required

`-constant`	Connect the object to a constant (0 or 1). This option will only work for hpin/pin/hport/port objects. When specified, any existing net connections to the *objects* specified will be disconnected prior to connecting to the constant. It does not connect the objects together, but instead adds a constant to each separate object at the level of hierarchy where the object exists. If an hpin or a pin is connected to an hnet (e.g. is not dangling), the constant is added to the hnet, otherwise it is added directly to the hpin/pin. If a constant is added to a port/hport, it is always added to the hnet of the port/hport. If there are existing drivers or constants attached to any net of the hnets mentioned above, an error message is given, but the connection is done anyway. *Data_type*: bool, optional
`-hnet` *`hnet_object`*	Specifies the hnet object to be used for the connections. It is mutually exclusive with `-net` (and `-constant`, `-net`). Additionally, the `-hnet` option can only be used to make a local connection (i.e. at the same level of hierarchy) *Data_type:* hnet, optional
`-net` *`net_object`*	Net object to be used for the connections. The object will be connected to the hnet of the specified net at the same level of hierarchy as the object. If there is no hnet at the same level of hierarchy, hports/hpins will be created using the base_name of the net object. Also, when `-net` is specified, any existing net connections to the *objects* specified will be disconnected prior to connecting to this net (for example, equivalent to running disconnect on each object first). *Data_type:* (net), optional
`-net_name` *`net_name`*	Net name to create when only pins specified. Net name to create when only pins specified (i.e. when -net is not specified). This is the base_name of the net and will replace the name of all nets connected to the pins in the object list. If a hierachical connection is being made the base_name will also be used to name any hports/hpins that are created. If *`net_name`* has any illegal chars, it will be automatically escaped. Thus -name "i1/n2" will result in "i1\/n2" for the .escaped_name, since it uses DEF escaping semantics. This is equivalent to checking if *`net_name`* is a legal Verilog identifier, and if not, prepend \ and add a trailing space to the internal Verilog name. *Data_type:* string, optional

Examples

The following command connects two pins with the given net name:

```
connect {pin:H1/I1/Y pin:H1/I3/A} -net_name H1/N1
```

Related Information

- disconnect

connect_hpin

```
connect_hpin
[-help]
-hinst moduleName
-net netName
[-no_new_hpins]
-pin_name hpinName
```

Attaches an hpin in the specified hinst (or top level design) to a net.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `connect_hpin` parameter. For a detailed description of the command and all of its parameters, use the man command: `man connect_hpin`
`-hinst` *hinstName*	Specifies the hinst to that has the hpin. To specify the top design, enter '`-`'. *Data_type*: string, required
`-net` *netName*	Specifies the name of the net. *Data_type:* string, required
`-no_new_hpins`	Prevents the creation of new hpins. If the hpin cannot connect to the net in a different hierarchy through existing hpins, the software displays an error message and the command stops running. *Default*: If you do not specify this parameter, the software creates new hpins as required to reach the net. *Data_type*: bool, optional
`-pin_name` *pinName*	Specifies the base_name of the hpin on the hinst. *Data_type*: string, required

Examples

- The `connect_hpin` command connects the hpin with base_name `p1` on hinst named h1/h2 to the net `h1/h2/net3`:

  ```
  connect_hpin -hinst h1/h2 -pin_name p1 -net h1/h2/net3
  ```

Related Information

- disconnect_hpin

connect_pin

```
connect_pin
[-help]
[-hport portName]
-inst instName
[-module verilogModule]
-net netName
[-no_new_port]
-pin termName
[-reference_pin refInstName refTermName]
```

Attaches a pin to a net. If the pin already connects to a different net, the software first detaches the pin from the current net, then attaches it to the new net.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `connect_pin` parameter. For a detailed description of the command and all of its parameters, use the man command: `man connect_pin`
`-inst` *`instName`*	Specifies the instance containing the pin. If the instance name does not exist, the software displays an error message and the command stops. *Data_type:* string, optional
`-module` *`verilogModule`*	Attaches the pin to the specified verilog module. *Data_type*: string, optional
`-net` *`netName`*	Specifies the name of the net to attach to the pin. If the net name does not exist, Innovus displays an error message and the command stops. *Data_type*: string, required
`-no_new_port`	Prohibits Innovus from creating hierarchical ports when it attaches a pin. If the pin cannot connect to the net through existing ports, Innovus displays an error message and the command stops. *Default:* If you do not specify this parameter, Innovus creates hierarchical ports as needed *Data_type*: bool, optional
`-pin` *`termName`*	Specifies the name of the pin. *Data_type*: bool, optional
`-reference_pin` *`refInstName refPinName`*	Specifies the pin *`refPinName`* on instance *`refInstName`* connected to the net that Innovus connects to the pin. You cannot specify the `-hport` parameter if you use this parameter. *Data_type*: string, optional
`-hport` *`portName`*	Specifies the hierarchical port used to connect the pin with the net. If you specify this parameter, the hierarchical port must exist in the module that contains the instance. The hierarchical port must connect to the net. This parameter lets you use a specific hierarchical port to maintain the same netlist topology, which simplifies equivalence checking later in the design flow. You cannot specify the -reference_pin parameter if you use this parameter. *Default:* If you do not specify this parameter, Innovus uses existing hierarchical ports or creates new hierarchical ports as necessary to connect the pin to the net. *Data_type*: string, optional

Examples

- The following command attaches pin `in1` of instance `i1/i2/i3 to net i1/i2/net26`:
  ```
  connect_pin -inst i1/i2/i3 -pin in1 -net i1/i2/net26
  ```
  If `i1/i2/i3` does not exist, or if `in1` is not a pin of `i1/i2/i3`, or if `i1/i2/net26` does not exist, the software displays an error message and the command stops.

- The following command attaches pin `in2` of instance `i1/i2/i3` to net `net27`, using hierarchical port `myPort`:
  ```
  connect_pin -inst i1/i2/i3 -pin in2 -net net27 -hport myPort
  ```
  The hierarchical port `myPort` must exist in the module definition for `i1/i2`, and `myPort` must already connect to `net27`. If this is not the case, the software displays an error message and the command stops.

- The following command attaches pin `in3` on instance `i1/i2/i3` by re-using existing hierarchical ports to `net28`:
  ```
  connect_pin -no_new_port -inst i1/i2/i3 -pin in3 -net net28
  ```
  No new hierarchical ports is allowed to be created to make the connections. If no existing hierarchical ports can be re-used to complete the connection, the software displays an error message and the command stops.

- The following command attaches pin Y of instance `testInst` to the Verilog module `hier_t3` using the net `testnet`:
  ```
  connect_pin -module hier_t3 -inst testInst -pin Y -net testnet
  ```

Related Information

- `disconnect_hpin`

create_diode

```
create_diode
[-help]
[-prefix prefix]
-diode_cell diodeCellName
-pin instName termName
[-location x y
[-orient r0 r90 r180 r270 mx mx90 my my90]]
```

Adds an antenna diode to a post-routed design. Innovus places the diode as close as possible to the specified pin.

When *instName* is specified, the new instance is bound with the correct cell in the power domain.

When `-location` is specified, the location also gives a power domain, which is checked against the power domain of the pin. If the power domains are different, a warning is issued.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `create_diode` parameter. For a detailed description of the command and all of its parameters, use the man command: `man create_diode`
`-diode_cell` *diodeCellName*	Specifies the name of the diode cell master. *Data_type*: string, required
`-location` *x y*	Specifies the location of the lower left corner of the diode. *Data_type*: string, optional
`-orient r0 r90 r180 r270 mx mx90 my my90`	Specifies orientation of the placed diode. To specify this parameter, you must also specify `-location`. *Data_type*: string, optional
`-pin` *instName termName*	Specifies the pin associated with the diode. *Data_type*: string, required
`-prefix` *prefix*	Specifies the prefix name of the diode cell instances. Innovus appends integers to differentiate the instances. *Data_type*: string, optional

Example

- The following example adds a diode of type `D1` for `term2` on instance `i2`, at location `500`, `600`:

```
create_diode -diode_cell D1 -pin i2 term2 -location 500 600
```

create_hinst

```
create_hinst
[-help]
[-parent module|hinst|design]
{-name hinst_name }
{-module module|design}
```

Adds a hierarchical module to the design.

You can use this command after importing a design.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `create_hinst` parameter. For a detailed description of the command and all of its parameters, use the man command: `man create_hinst`
`-module module|design`	Specifies the module to use for the new hinst. The new hinst will have hports/hpins created based on the module ports as well as hnets of the same name. *Data_type:* (module|design), required
`-name hinst_name`	Specifies the name of the new hinst. If this is the full hierarchical path, then the parent will be derived from the name (i.e. the parent name is the string before the last "/". In this case, the base_name of the new hinst (i.e. the string after the last "/" should not have any special characters that require escaping. When the `-parent` option is used, then the name is the base_name of the new hinst. if the name has any illegal chars, it will be automatically escaped. Thus -name "i1/i2" will result in "i1\/i2" for the .escaped_name, since it uses DEF escaping semantics. This is equivalent to checking if <hinst_name> is a legal Verilog identifier, and if not, prepend \ and add a trailing space to the internal Verilog name. *Data_type*: string, required
`-parent module|hinst|design`	Specifies the parent for the new hinst. It supports both the object or the name for the specified object types. *Default*: Current design *Data_type*: (hinst|design|module), optional

Examples

The following example creates the module `module:carve/mod1` and instantiates it:

```
@innovus 1> create_module -name mod1
module:carve/mod1
@innovus 2> create_hinst -name H1 -module mod1
hinst:carve/H1
```

Related Information

- create_module
- delete_hinst

create_hnet

```
create_hnet
[-help]
[-bus msb:lsb]
{-name hnet_name}
[-parent module|hinst|design]
[-power | -ground ]
```

Creates a new hnet (and parent net) object and returns the newly created hnet object pointer.

Parameters

`-bus` *`msb:lsb`*	Specifies the bit range for bus nets. You must separate the start and end IDs with a colon (:). If there is an existing bus (for example, 3:0) then additional bits can be added provided they are adjacent to the existing bits (7:4 for example). By default, the tool creates a scalar hnet. *Data_type*: string, optional
`-ground`	Creates a ground net. *Data_type*: bool, optional
`-help`	Outputs a brief description that includes the type and default information for each `create_hnet` parameter. For a detailed description of the command and all of its parameters, use the man command: `man create_hnet`
`-name` *`hnet_name`*	Specifies the name of the new hnet. If this is the full hierarchical path, then the parent will be derived from the name (i.e. the parent name is the string before the last "/". In this case, the base_name of the new hnet (i.e. the string after the last "/" should not have any special characters that require escaping. When the `-parent` option is used, the name is the base_name of the new hnet. If the name has any illegal chars, it will be automatically escaped. Therefore, `-name "i1/n2"` will result in "i1\/n2" for the .escaped_name, as it uses DEF escaping semantics. This is equivalent to checking if *`hnet_name`* is a legal Verilog identifier, and if not, prepend \ and add a trailing space to the internal Verilog name. *Data_type*: string, required
`-parent` *`module|hinst|design`*	Specifies the parent for the new hinst. It supports both the object or the name for the specified object types, *`module`*, *`hinst`*, *`design`*. *Default*: Current design *Data_type*: (hinst|design|module), optional
`-power`	Creates a power net. *Data_type*: bool, optional

Example

```
CPU
 |
 +-----------------+
```

```
 |                |
 H1 (alu)         H2 (fsm)
 |
 +----------------+
 |                |
+ H1 (add)        H2 (mult)
```

- The following command creates a scalar top level net/hnet named N1

  ```
  create_hnet -name N1
  ```

- The following command creates a 8-bit net/hnet named H1/H1/B1

  ```
  create_hnet -name B1 -bus 7:0 -parent module:add
  ```

- The following command creates an 8-bit net/hnet named H1/H1/B1

  ```
  create_hnet -name B1 -bus 7:0 -parent H1/H1
  ```

- The following commands add additional bits to an existing hnet:

  ```
  get_db hnets *CN1*
  {hnet:cpu/CH1/CN1[1]} {hnet:cpu/CH1/CN1[0]}
  create_hnet -parent CH1 -name CN1 -bus 4:2
  {hnet:cpu/CH1/CN1[4]} {hnet:cpu/CH1/CN1[3]} {hnet:cpu/CH1/CN1[2]} {hnet:cpu/CH1/CN1[1]}
  {hnet:cpu/CH1/CN1[0]}
  ```

create_hport

```
create_hport
[-help]
[-bus msb:lsb]
-direction {in out inout}
-name hport_name
[-parent module|hinst|design]
```

Create new hports and corresponding hpins for the parent object. It also implicitly creates hnet and net objects inside the parent object with the same name as the new hport. If the new hport is a bus, then the bus_bits objects are also created. The hnet object has no corresponding hnet or net object but is left dangling. It returns the newly created hport object pointer.

Parameters

`-bus msb:lsb`	Specifies the bit range for bus hports. You must separate the start and end IDs with a colon (:). If there is an existing bus (for example, 3:0) then additional bits can be added provided they are adjacent to the existing bits (7:4 for example). By default, the tool creates a scalar hport. *Data_type*: string, optional
`-direction {in out inout}`	Specifies the direction of the new port, in, out, inout. *Data_type*: enum, required
`-help`	Outputs a brief description that includes the type and default information for each `create_hport` parameter. For a detailed description of the command and all of its parameters, use the man command: `man create_hport`
`-name hport_name`	Specifies the name of the new hport. If this is the full hierarchical path, then the parent will be derived from the name (i.e. the parent name is the string before the last "/". In this case, the base_name of the new hport (i.e. the string after the last "/") should not have any special characters that require escaping. When the `-parent` option is used, then the name is the base_name of the new hport. If the name has any illegal chars, it will be automatically escaped. Therefore, `-name "hp1/2"` will result in `"hp1\/2"` for the .escaped_name, since it uses DEF escaping semantics. This is equivalent to checking if *hport_name* is a legal Verilog identifier, and if not, prepend \ and add a trailing space to the internal Verilog name. *Data_type*: string, required
`-parent module|hinst|design`	Specifies the parent for the new hport. *Data_type*: (hinst|design|module), optional

Example

```
CPU
 |
 +----------------+
 |                |
 H1 (alu)         H2 (fsm)
 |
 +----------------+
 |                |
+ H1 (add)        H2 (mult)
```

- The following command creates new scalar objects `hpin:CPU/H1/H1/P1`, `hport:CPU/H1/H1/P1`, and `hnet/net H1/H1/P1`.

  ```
  create_hport -name P1 -parent add -direction in
  ```

- The following command creates a new 8-bit objects `hpin:H1/H2/BP1`, `hport:H1/H2/BP1`, and `hnet/net H1/H2/BP1`.

  ```
  create_hport -name BP1 -parent H1/H2 -bus 7:0
  ```

- The following command creates new scalar objects `hpin:CPU/H1/H2/P2`, `hport:CPU/H1/H2/P2`, and `hnet/net H1/H1/P2`.

  ```
  create_hport -name H1/H2/P2 -direction out
  ```

- The following commands add a single bit to an hport:

  ```
  get_db hports *data_in*
  {hport:cpu/H1/data_in[3]} {hport:cpu/H1/data_in[2]} {hport:cpu/H1/data_in[1]}
  {hport:cpu/H1/data_in[0]}
  create_hport -parent H1 -name data_in -bus 4:4 -direction in
  ```

create_inst

```
create_inst
[-help]
[-dont_snap_to_placement_grid]
[-location {x y}]
[-orient {r0 r90 r180 r270 mx mx90 my my90}]
[-physical]
{-base_cell base_cell} {-name inst_name}
[-parent module|hinst|design]
[-place_status place_status]
```

Creates a new instance object, optionally places it in the design, and returns the newly created object pointer. You can run this command after the design is in memory.

For MSV designs, a cell specified using the `-base_cell` parameter should exist in libraries bound to the power domain of the newly added inst. When `-location` is specified, the location is checked for a power domain, which is checked against the power domain of the parent. If the power domains are different, a warning is issued.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `create_inst` parameter. For a detailed description of the command and all of its parameters, use the man command: `man create_inst`
`-physical`	Places a physical instance without updating the netlist. *Data_type*: bool, optional
`-base_cell base_cell`	Specifies the base_cell to use for the new instance. *Data_type*: (base_cell), required
`-dont_snap_to_placement_grid`	If `-dont_snap_to_placement_grid` is specified, Innovus places the instance at the specified location even if the location is off the placement grid. This parameter is used with `-location`. *Default*: false *Data_type*: bool, optional

`-name` *inst_name*	Specifies the name of the new inst. If this is the full hierarchical path, then the parent will be derived from the name (i.e. the parent name is the string before the last "/". In this case, the base_name of the new inst (i.e. the string after the last "/") should not have any special characters that require escaping. When the `-parent` option is used, then the name is the base name of the new inst. If the name has any illegal characters, it will be automatically escaped. Therefore, `-name "i1/i2"` will result in `"i1\/i2"` for the .escaped_name, since it uses DEF escaping semantics. This is equivalent to checking if *inst_name* is a legal Verilog identifier, and if not, prepend \ and add a trailing space to the internal Verilog name. *Data_type*: string, required
`-location {`*x y*`}`	Specifies the location of the placed instance. *Default:* If you do not specify a location, Innovus places the instance at the design origin. *Data_type*: point, optional
`-orient {r0 | r90 | r180 | r270 | mx | mx90 | my | my90}`	Specifies orientation of the placed instance. *Default:* If you do not specify an orientation, Innovus uses `r0`. *Data_type*: enum, optional
`-parent` *module|hinst|design*	Specifies the parent for the new instance. It supports both the object or the name for the specified object types. Only honored for logical instances and is ignored when used with `-physical`. Default: Current design *Data_type*: (hinst|design|module), optional

`-place_status` *place_status*	Specifies the placement status of the added instance. You can specify the following as the placement status: • `cover`: Specifies that the instance has a location and is a part of a cover macro. It cannot be moved by automatic tools or interactive commands. You must specify the instance's location and its orientation. • `placed`: The status of the added instance is placed. • `fixed`: The status of the added instance is fixed. • `soft_fixed`: The status of the added instance is soft_fixed. It cannot be moved by global placement and can only be moved by the legalization step of detail placement • `unplaced`: The status of the added instance is unplaced. *Default*: `unplaced` *Data_type*: enum, optional **Note:** This option is used along with the `-location` option. It should not be used along with the `-parent` option.

Examples

- The following command creates `inst:<design>/H1/I1`.
  ```
  create_inst -name I1 -parent H1 -base_cell BUF1
  ```
- The following command adds buffer instance `i1/i2` at location `100, 200`:
  ```
  create_inst -base_cell BUFX1 -name i1/i2 -location {100 200}
  ```
- The following command adds buffer instance `insta` to the verilog module `HIER_2`:
  ```
  create_inst -parent HIER_2 -base_cell BUFX2 -name insta
  ```

Related Information

- `delete_inst`

create_module

```
create_module
[-help]
-name module_name
```

Create new hports and corresponding hpins for the parent object. It also implicitly creates hnet and net objects inside the parent object with the same name as the new hport. If the new hport is a bus, then the bus_bits objects are also created. The hnet object has no corresponding hnet or net object but is left dangling. It returns the newly created hport object pointer.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `create_module` parameter. For a detailed description of the command and all of its parameters, use the man command: `man create_module`
`-name` *`module_name`*	Specifies the name of the new module. If the name has any illegal chars, it will be automatically escaped. Therefore, `-name "m1/m2"` will result in `"m1\/m2"` for the .escaped_name, since it uses DEF escaping semantics. This is equivalent to checking if *`hinst_name`* is a legal Verilog identifier, and if not, prepend \ and add a trailing space to the internal Verilog name. *Data_type*: string, required

Example

- The following command creates `module:<design>/mod1`.

  ```
  create_module -name mod1
  ```

- The following command creates `hinst:<design>/H1`.

  ```
  create_hinst -name H1 -module mod1
  ```

- The following command creates `hpin:<design>/H1/in`.

  ```
  create_hport -name in -module mod1 -input
  ```

- The following command creates `hpin:<design>/H1/out`.

  ```
  create_hport -name out -module mod1 -output
  ```

- The following command creates `inst:<design>/H1/I1`.

  ```
  create_inst -name I1 -parent H1 -base_cell BUF1
  ```

create_net

```
create_net
[-help]
[-bus startID:endID]
[-module verilogModule]
-name netName
[-power  | -ground ]
```

Creates a flat physical net, that is the parent is always the design. The net can optionally be tagged as a power or ground net. The .is_physical_only attribute will be set to `true` along with .is_power or .is_ground when specified. The net is not logical and will not be written as part of the verilog netlist unless `write_netlist -phys` is specified. Currently only pins and PG pins are supported. If a pin at any level of hierarchy is connected to a power or ground physical net, a local 1'b0 or 1'b1 will be connected to the pin at the same level of hierarchy as the pin, that is no hports or hpins will be created as it is treated as a global physical net.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `create_net` parameter. For a detailed description of the command and all of its parameters, use the man command: `man create_net`
`-bus` *`startID`*`:`*`endID`*	Specifies the bit range for bus nets. The *`startID`* and *`endID`* indicate the first and last bits on the bus. You must separate the start and end IDs with a colon (:). If the specified bus does not exist, tool will create a bussed Verilog net. If the specified bus already exists, tool will do bus net resizing. The specified bus bit range must be adjacent to the first or last bit of the existing bus. Also, it must have the same ascending or descending order as the existing bus. **Note**: PG bus net is not supported *Data_type*: string, optional
`-ground`	Adds a ground net. *Data_type*: bool, optional
`-module` *`verilogModule`*	Adds the new net to the specified verilog module. If the module name contains special characters, you need to start the module name with a backlash '\' and end it with a space. *Data_type*: (module|design), optional
`-name netName`	Specifies the net to add. If the net name contains special characters, you need to start the part of the net name, which contains special characters, with a backlash '\' and end it with a space. *Data_type*: string, required **Note:** You must specify *`-name`* before you specify other parameters. The software issues an error message if you do not specify the *`-name`* first.
`-power`	Adds a power net. *Data_type*: bool, optional

Examples

- The following command adds net `i1/n1`:

  ```
  create_net i1/n1
  ```

- The following command adds one bit i1/n1[6] to existing bus i1/n1[2:5]:

```
create_net -name i1/n1 -bus 6:6
```

- The following command adds three bits i1/n1[-1], i1/n1[0], i1/n1[1] to existing bus i1/n1[2:5]:
  ```
  create_net -name i1/n1 -bus -1:1
  ```
- The following command adds net with base name ant_net to hierarchy `DTMF_INST/TDSP_CORE_INST/TDSP_CORE_MACH_INST[0].g`:
  ```
  create_net {DTMF_INST/TDSP_CORE_INST/\TDSP_CORE_MACH_INST[0].g /ant_net}
  ```
  Here the "[" and "]" brackets between the backlash and the space are special characters.
- The following command adds net with base name `TDSP_CORE_INST/TDSP_CORE_MATCH_INST[0].g/ant_net` to hierarchy `DTMF_INST`:
  ```
  create_net {DTMF_INST/\TDSP_CORE_INST/TDSP_CORE_MATCH_INST[0].g/ant_net }
  ```
- The following command adds net with base name `ant_net[1]` to `module H[1]`:
  ```
  create_net {\ant_net[1] } -module {\H[1] }
  ```

Related Information

- `delete_nets`
- `write_netlist`

create_port

```
create_port
[-help]
[-bus msb:lsb]
-direction {in out inout}
-name port_name
```

Create new ports for a design object and returns the newly created port object pointer. It also implicitly creates hnet and net objects with the same name connected to the port.

Parameters

`-bus msb:lsb`	Specifies the bit range for bus ports. You must separate the start and end IDs with a colon (:). If there is an existing bus (for example, 3:0) then additional bits can be added provided they are adjacent to the existing bits (7:4 for example). Default is to create a scalar port. *Data_type*: string, optional
`-direction {in out inout}`	Specifies the direction of the new port, in, out, inout. *Data_type*: enum, required
`-help`	Outputs a brief description that includes the type and default information for each `create_module` parameter. For a detailed description of the command and all of its parameters, use the man command: `man create_module`. *Data_type*:
`-name module_name`	Specifies the name of the new port. If the name has any illegal chars, it will be automatically escaped. Therefore, `-name "p1/2"` will result in `"p1\/2"` for the .escaped_name, since it uses DEF escaping semantics. This is equivalent to checking if *port_name* is a legal Verilog identifier, and if not, prepend \ and add a trailing space to the internal Verilog name. *Data_type*: string, required

Example

- The following command creates a new input scalar port called P1:
  ```
  create_port -name P1 -direction in
  ```
- The following command creates a new 8-bit output port called BP1
  ```
  create_port -name BP1 -bus 7:0 -direction out
  ```

delete_assigns

```
delete_assigns
[-help]
[-add_buffer [-buffer_tie_assigns] ]]
[-buffer_name bufName]
[-honor_preserve]
[-ignore_port_constraints]
[-prefix bufNamePreFix]
[-nets netName [-net_buffer_name netBufName]]
[-report]
```

Removes assign statements from the Verilog as much as possible while keeping the same DEF netlist. This means that the placement and routing remain the same and timing is not disturbed.

The `delete_assigns` command removes all non-top module assign nets without adding buffers.There are cases where you do not want assign nets in your design. However, there are some assign nets in the original net list imported into Innovus, or you have a design with some unwanted assign nets. In such cases, you can use `delete_assigns` to remove all assigns or replace assigns with proper buffers that are automatically selected by the tool.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `delete_assigns` parameter. For a detailed description of the command and all of its parameters, use the man command: `man delete_assigns`
`-add_buffer`	Used if assign net buffering feature is needed for command `delete_assigns`. *Data_type*: bool, optional
`-buffer_name bufName`	Specifies the name of the buffer for all assign nets, disables automatic buffer selection. *Data_type*: string, optional
`-buffer_tie_assigns`	Allows to buffer tie assigns. *Data_type*: bool, optional
`-honor_preserve`	Honors the preserve setting while doing the assignment removal. *Data_type*: bool, optional
`-ignore_port_constraints`	Allows to ignore all port constraints during assign removal. *Data_type*: bool, optional

`-nets` *netName*	Specifies the name of net (s). Supports removing or buffering assigns of a list of assign nets,
	For example, the following command removes all assigns of assign nets n1, n2 and n3.
	`delete_assigns -nets {n1 n2 n3}`
	The following command replaces all assigns of assign nets n1, n2 and n3 with buffers.
	`delete_assigns -add_buffer -nets {n1 n2 n3}`
	If "-net" is used, "-buffering" must also be used for buffering those assign nets specified in *netName*. Otherwise, `delete_assigns` will work same as no options are specified (as default).
	Whenever option "-net" is used, *netName* must be provided. Otherwise, a warning message is issued.
	You can specify any of the following as the *netName*:
	• Single net name
	• Multiple net names separated with space
	• Single wildcard name (all matched assign nets will be collected for buffering)
	• Multiple wildcard names separated with space (all match assign nets will be collected for buffering)
	• Single bus name
	• Multiple bus names
	Note: For each bus, all or portion of bus bits can be assign nets and the others can be non-assign nets. The buffering code replaces only those assign bus bits while leave those non-assign bus bits unchanged.
	Data_type: string, optional

`-net_buffer_name` *netBufName*	Specifies the name of the buffer for the specific net given in the `-nets` option. This option, if specified, should come after the `-nets` option. *Data_type*: string, optional
`-prefix` *bufNamePreFix*	Specifies the user-specified prefix for buffer names. If `-prefix` is used, you need to specify a prefix `bufNamePreFix` for buffer names that may be used in this command. The default prefix for added buffers is "`buf_dummy_`" if you do not specify it. *Data_type*: string, optional

`-report`	Reports list of all buffers and assign nets that were replaced by the buffers. If `-report` is used, a list of all buffers and assign nets that were replaced by the buffers will be reported in log file. If buffers are inserted to the netlist you may need to use Innovus to place and route them. *Data_type*: bool, optional

The `delete_assigns` command:

- Removes assigns that involve only internal "wires".
  For example, in the code below, `assign n2 = n1` is removed.

```
module top();
  wire n1, n2;
  assign n2 = n1;

  BUFX2 u1(.Y(n1));
  BUFX2 u2(.A(n2));
  endmodule
```

  becomes

```
module top();
  wire n1;

  BUFX2 u1(.Y(n1));
  BUFX2 u2(.A(n1));
  endmodule
```

- Removes assigns that involve only one port.
  For example, in the code below, `assign p1 = n1` is removed.

```
  module hier(p1);
  output p1;
  wire n1;
  assign p1 = n1;
   BUFX2 u1(.Y(n1));
  endmodule
```

becomes

```
module hier(p1);
output p1;
 BUFX2 u1(.Y(p1));
endmodule
```

- Rewires nets with feedthrough assigns that involve at least two ports, at some higher level:

```
module hier2 (
          i,
          o);
     input i;
     output o;
     assign o = i ;
     BUFX2 u1 (.A(i));
  endmodule
  module hier (
          i,
          o);
     input i;
     output o;

     hier2 u1 (.i(i),
          .o(o));
  endmodule
  module top ();
     // Internal wires
     wire i;
     wire o;
     BUFX2 u1 (.Y(i));
     hier u2 (.i(i),
          .o(o));
```

```
    BUFX2 u3 (.A(o));
  endmodule
```

becomes

```
module hier2 (
          i,
          o);
    input i;
    output o;

    BUFX2 u1 (.A(i));
  endmodule
  module hier (
          i,
          o);
    input i;
    output o;
    hier2 u1 (.i(i));
  endmodule
  module top ();
    // Internal wires
    wire i;

    BUFX2 u1 (.Y(i));
    hier u2 (.i(i));
    BUFX2 u3 (.A(i));
  endmodule
```

delete_bus_net

```
delete_bus_net
[-help]
[-module_based moduleName]
{-bus busName | -bus_from_bit busBitName}
```

Deletes bus net from a design.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `delete_bus_net`. For a detailed description of the command and all of its parameters, use the man command: man `delete_bus_net`
`-module_based moduleName`	Specifies the module name that contains the bus net which needs to be deleted. *Data_type*: string, optional
`-bus busName`	Specifies the base name of the top-level bus net, or the base name of a bus net in verilog module when used with `-module_based` option. *Data_type*: string, optional
`-bus_from_bit busBitName`	Specifies the bus bit name of a top level or a non-top level bus net, or a bus bit name of a bus net in verilog module when used together with the `-module_based` option. *Data_type*: string, optional

⚠ The `delete_bus_net` command does not support wildcards.

Examples

Non-top level bus net

- The following command deletes the bus net x/y/z.

  ```
  delete_bus_net -bus_from_bit x/y/z[2]
  ```

- The following command deletes a bus net from a verilog module, used with `-module_based` and `-bus_from_bit` *busBitName*.

  ```
  delete_bus_net -bus_from_bit z[2] -module_based Y
  ```

- The following command deletes a bus net from a verilog module, used with `-module_based` and `-bus` *busName*.

  ```
  delete_bus_net -bus z -module_based Y
  ```

Top level bus net

- The following command deletes the top-level bus net, using the option `-bus_from_bit` *busBitName*.

  ```
  delete_bus_net -bus_from_bit x[2]
  ```

- The following command deletes the top-level bus net, using the option `-bus` *busName*.

  ```
  delete_bus_net -bus x
  ```

delete_empty_hinsts

```
delete_empty_hinsts
-help
[-delete_empty_1801_hinsts]
```

Deletes modules and their instantiations. Modules and their instantiations are deleted if the following criteria are met:

- The netlist is unique
- The module does not contain any instances, or the module contains only unconnected instances of other empty modules
- The instances of the module are not connected to other instances or ports

Note: The `delete_empty_hinsts` command deletes only those modules that are below the top-level design module.

The `delete_empty_hinsts` command does not delete used empty modules, where there exists at least one connected instance of the empty cell. For example, black boxes typically are empty, but they are still connected to other instances or ports.

At times, the lowest hierarchy is connected to the immediate higher-level hierarchy but the module is dangling and should be deleted. You can delete the dangling modules by first deleting the dangling ports followed by the deletion of the empty modules as given below:

```
delete_dangling_ports

delete_empty_hinsts
```

Note: A design can contain empty modules for different reasons. Modules can become empty due to optimization processes. A design also can contain MCPO modules that were read in with the netlist. To convert empty modules into special leaf cells when reading in the netlist, see `init Category Attributes`.

Parameters

`-delete_empty_1801_hinsts`	Deletes empty UPF modules. *Data_type*: bool, optional
`-help`	Outputs a brief description that includes the type and default information for each `delete_empty_hinsts` parameter.

Related Information

- delete_dangling_ports
- init Category Attributes

delete_feedthru_buffer

```
delete_feedthru_buffer
[-help]
[-dont_delete_dangling_hports]
[-non_preferred_keep_nets netNamePattern] [-preferred_keep_nets netNamePattern]
{{-inst bufInstName} | {-inst_prefix instPrefixName}
```

Removes the feedthrough buffer specified in the `-inst` option. It also removes all possible dangling horts/hnets that are associated with the buffer removed. The `delete_feedthru_buffer` command supports multiple-instantiated buffer cells.

Parameters

`-inst bufInstName`	Specifies the name of the feedthrough buffer to be removed. *Data_type*: string, optional
`-dont_delete_dangling_hports`	Is used if you do not want to remove the dangling ports created by this command. *Data_type*: bool, optional
`-help`	Outputs a brief description that includes the type and default information for each `delete_feedthru_buffer` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man delete_feedthru_buffer`
`-inst_prefix` *instPrefixName*	Specifies the prefix name for the feedthrough buffer. It removes all buffer instances that were created by the `add_partition_feedthrus` command if `add_partition_feedthrus` also used this option to specify the exactly same `instPrefixName`. Therefore, you need to specify exactly the same instance name prefix in the `-instPrefix` option of both the commands. *Data_type*: string, optional **Note**: The `-instPrefix` specified here must be unique to all instance names in the current design.
`-non_preferred_keep_nets` *netNamePattern*	Specifies the non-preferred keep net name pattern. The nets whose names match the pattern will not be used by buffer deletion. *Data_type*: string, optional
`-preferred_keep_nets` *netNamePattern*	Specifies the preferred keep net name pattern. The nets whose names match the pattern will be used by buffer deletion. *Data_type*: string, optional

Example

- The following command deletes feedthrough buffer with inst name as `i1/i2/FE_FEED_BUF_1`, and also keeps the dangling ports:
  ```
  delete_feedthru_buffer -inst i1/i2/FE_FEED_BUF_1 -dont_delete_dangling_hports
  ```

Related Information

- add_partition_feedthrus

delete_hinst

```
delete_hinst
[-help]
[-module | -modules ]
hinst_name+
```

Deletes a hierarchical instance and its module and optionally, its cell.

Parameters

`-help`	Prints a brief description that includes the type and default information for each `delete_hinst` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man delete_hinst`.
`hinst_name+`	Specifies the hierarchical instance to delete. *Data_type*: (hinst)+, required
`-module`	Deletes the module of the hierarchical instance. It only works when the specified hierarchical instance is the last one in the design that references this module. *Data_type*: bool, optional
`-modules`	Deletes all instances that are hierarchically inside the instance from which the given hierarchical instance, *`hinst_name`*, is instantiated. If a hierarchical instance is multi-referenced and all sibling hierarchical instances instantiated from the hierarchical instance are hierarchically inside *`hinst_name`*, the hierarchical instance will be deleted. Otherwise, the multi-referenced hierarchical instance will not be deleted. *Data_type*: bool, optional

Examples

- The following command deletes instance RESULTS_CONV_INST:
  ```
  delete_hinst DTMF_INST/RESULTS_CONV_INST
  ```

- The following command deletes instance RESULTS_CONV_INST and deletes its cell:
  ```
  delete_hinst DTMF_INST/RESULTS_CONV_INST -module
  ```

Related Information

- create_hinst

delete_hpin

```
delete_hpin
[-help]
[-bus startId:endId]
-hinst moduleName
-pin_name pinName
```

Disconnects the specified port from its net and deletes the port. Individual bits of a bus port cannot be deleted.

Parameters

`-bus startId:endId`	Specifies the startId and the endId of the bus bit. *Data_type*: string, optional **Note**: • When a bit port or bit net is connected to something else, the bit will not be removed. • Bus bit ports to be deleted must be adjacent to each other and adjacent to either msb or lsb of the existing bus.
`-help`	Outputs a brief description that includes the type and default information for each `delete_hpin` parameter. For a detailed description of the command and all of its parameters, use the man command: `man delete_hpin`
`-hinst moduleName`	Specifies the name of the hierarchical instance whose port is to be deleted. Specify "-" for the top hierarchical instance. *Data_type*: string, required
`-pin_name pinName`	Specifies the name of the port to be deleted. Wildcards are not accepted. *Data_type*: string, required

Examples

- The following command deletes the port 'rst_i' from the top hierarchical instance:
  ```
  delete_hpin -hinst - -pin_name rst_i
  ```
- The following command deletes all bits of the bus port '`data_i`' of the hierarchical instance `moduleB_inst`:
  ```
  delete_hpin -hinst moduleA_inst/moduleB_inst -pin_name data_i
  ```
- The following command deletes bus bits out[7], out[6], out[5], and out[4] from existing top-level bus port out[7:0]:
  ```
  delete_hpin -hinst - -pin_name out -bus 7:4
  ```

Related Information

- `create_hport`

delete_inst

```
delete_inst
[-help]
-inst instName
[-module moduleName ]
[-verbose]
```

Deletes instances from the current design. Innovus checks whether the instance is physical or logical, and deletes the instance accordingly.

Note: The `delete_inst` command cannot be used in the post-mask ECO. Please use `read_eco <eco_files> -post_mask` to delete instances in the post-mask ECO.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `delete_inst` parameter. For a detailed description of the command and all of its parameters, use the man command: `man delete_inst`
`-inst` *instName*	Specifies the list of instances to delete. Does not accept *inst* dual ported objects. *Data_type*: string, required
`-module` *moduleName*	Specifies the name of the Verilog module. *Data_type*: string, optional
`-verbose`	Prints out the deleted instance message. *Data_type*: bool, optional

Example

- The following command deletes instance `i1/i2`:

  ```
  delete_inst -inst i1/i2
  ```

- The following command deletes all instances whose base names begin with `i` inside hierarchical instance `i1`:

  ```
  delete_inst -inst i1/i*
  ```

- The following command deletes the instance `insta` from the verilog module `HIER_2`:

  ```
  delete_inst -module HIER_2 -inst insta
  ```

- The following command deletes all instances in the `SPGM` module:

  ```
  delete_inst -inst * -module SPGM
  ```

Related Information

- `create_inst`
- `delete_obj`
- `delete_filler`
- `eco_batch_mode` in `eco Category Attributes`
- `read_eco`

delete_nets

```
delete_nets
[-help]
net
[-bus startId:endId]
[-module verilogModule]
```

Logically deletes nets from a design. If a net is routed, wire segments associated with the net are also deleted.

If you plan to call `route_eco -route_only_layers bottomLayer:topLayer` later, ensure that the net to be deleted only has wire segments on the specified layer.

Note: To delete wire segments associated with a net without deleting it logically, use the `delete_routes` command.

You can use wildcards (`*?`) to specify the nets you want Innovus to delete.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `delete_nets` parameter. For a detailed description of the command and all of its parameters, use the man command: `man delete_nets`
`-bus startId:endId`	Deletes the specified bus bits. The specified bus bit net must be a one pin net, which means that the bus bit net is floating on the other side. The specified range must be adjacent to the most significant bit or the least significant bit of the existing bus. Also, it must have the same ascending or descending order as the existing bus. **Note**: PG bus net is not supported. *Data_type*: string, optional
`-module verilogModule`	Deletes the net from the specified verilog module. *Data_type*: string, optional
net	Specifies the name of the net to delete. *Data_type*: string, required

Example

- The following command deletes `net1`:

  ```
  delete_nets net1
  ```

- The following command deletes all nets whose name begin with `net1`:

  ```
  delete_nets net1*
  ```

- The following command deletes one bit h1/net2[3] from existing bus h1/net2[3:0]:

  ```
  delete_nets h1/net2 -bus 3:3
  ```

- The following command deletes two bits h1/net2[3], h1/net2[2] from existing bus h1/net2[3:0]:

  ```
  delete_nets h1/net2 -bus 3:2
  ```

Related Information

- `create_net`
- `delete_routes`
- `route_eco`

delete_notch_fill

`delete_notch_fill`

Deletes all gap, notch, and hole geometries.

- A gap is a minimum spacing violation with two edges without a common adjacent side.
- A notch is a minimum spacing violation with two edges with a common adjacent side.
- A hole is a minimum enclosure violation.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `delete_notch_fill` parameter.

Related Information

- ECO Flows chapter in the *Innovus Stylus Common UI User Guide*

disconnect

```
disconnect
[-help]
pin|hpin|port|hport|pg_pin+
[-keep_dangling_net]
```

Disconnects pins, ports, hports, or hpins from any hnet or net. For example, if A, B, and C are connected to net N , and you disconnect A, then B and C remain connected to N, but A is now left unconnected. For hpin/hport objects, they are disconnected only from the local hnet they are connected to. So, for an hpin, it will be disconnected from the hnet above the hpin (outside the hinst of the hpin). For hports, it will be disconnected from the hnet below the hport (the hnet inside the hinst of the hport) but the associated hpin/hport to hnet connection is not affected. You need to disconnect them both if you want both connections removed. Also, if an hpin/pin is directly connected to a constant (e.g. has .constant 0/1 and no hnet), the .constant attribute on the hpin/pin will be cleared, and return “no_constant”.

By default, dangling hnets that have no port, hport, hpin, or pin attached to it are deleted. Any constants like 1’b0/1’b1 are ignored, so a constant hnet is considered dangling if there are no port, hport, hpin or pin objects attached to it.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `disconnect` parameter. For a detailed description of the command and all of its parameters, use the man command: `man disconnect`
`-keep_dangling_net`	Keeps the net object even if it has no loads or drivers. Default is to delete dangling nets. *Data_type*: bool, optional
pin|hpin|port|hport|pg_pin+	List of objects to disconnect (port/pin/hpin/hport/pg_pin). Only objects are supported, no simple names. *Data_type*: (pin |hpin |port |hport |pg_pin)+, required

Example

Consider the following example:

```
@innovus1>get_db hport:H1/B1 .hnet

hnet:H1/B1

@innovus2>get_db hpin:H1/B1.hnet
```

```
hnet:P1
```

If the following command is run, the tool will disconnect hport:H1/B1 from hnet:H1/B1, but leave hpin:H1/B1 connected to hnet:P1

```
@innovus2>disconnect hport:H1/B1
```

Related Information

- connect

disconnect_hpin

```
disconnect_hpin
[-help]
-hinst hinstName
-pin_name portName
```

Detaches the net connected to the specified port on the specified hierarchical instance.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each disconnect_hpin parameter. For a detailed description of the command and all of its parameters, use the man command: `man disconnect_hpin`
`-hinst hinstName`	Specifies the name of the hierarchical instance. To detach a top-level port, specify "-" as the *hinstName*. For example, the following command detaches the port 'a' from the net connected to it: `>disconnect_hpin -hinst - -pin_name a` *Data_type*: string, required
`-pin_name portName`	Specifies the port on the module. *Data_type*: string, required

Example

- The following command detaches port `p1` from `moduleA`:
  ```
  disconnect_hpin -hinst moduleA -pin_name p1
  ```

Related Information

- connect_hpin

disconnect_pin

```
disconnect_pin
[-help]
[-module verilogModule]
-inst instName
-pin
[-net]
```

Disconnects a pin from a net.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `disconnect_pin` parameter. For a detailed description of the command and all of its parameters, use the man command: `man disconnect_pin`
`-inst instName`	Specifies the instance that contains the pin you want to detach. *Data_type*: string, required
`-module verilogModule`	Detaches pin of the specified instance inside the Verilog module. *Data_type*: string, optional
`-net`	Specifies the net that is already connected to the pin. If the pin does not connect to the specified net, or if the net does not exist, the software displays an error message and the command stops. The software uses this parameter for error checking only. *Data_type*: string, optional
`-pin`	Specifies the pin to disconnect. If the pin does not exist on the specified instance, the software displays an error message and the command stops. *Data_type*: string, required

Example

- The following command disconnects pin `in1` of instance `i1/i2/i3`:

  ```
  disconnect_pin -inst i1/i2/i3 -pin in1 -net i1/i2/net26
  ```

  If instance `i1/i2/i3` does not exist, or `in1` is not a pin of `i1/i2/i3`, the software displays an error message and the command stops. The net `i1/i2/net26` is specified, so if net `i1/i2/net26` does not exist, or if the pin is not already connected to net `i1/i2/net26`, the software displays an error message and the command stops.

Related Information

- `connect_pin`

eco_add_repeater

```
eco_add_repeater
[-help]
{-net netName | -pins {term1 term2} }
[-hinst_guide hinstName | -spread_distance distance| -spread_count number ]
[-name instName | -spread_distance distance | -spread_count number ]
[-spread_distance distance | -spread_count number ]
[-new_net_name netName | -spread_distance distance | -spread_count number ]
[-location {xLoc yLoc} | { xLoc yLoc inv_xLoc inv_yLoc} | -off_load_at_location {x1a y1a x1b y1b
x2a y2a x2b y2b ....} | { xLoc yLoc inv_xLoc inv_yLoc} | -off_load_slack slack | -spread_distance
distance | -spread_count number ]
[-radius radius | -logical_change_only | -spread_distance distance | -spread_count number ]
[-off_load_at_location {x1a y1a x1b y1b x2a y2a x2b y2b ....} | { xLoc yLoc inv_xLoc inv_yLoc} | -
off_load_slack slack | -spread_distance distance | -spread_count number ]
[{[-location {xLoc yLoc} | { xLoc yLoc inv_xLoc inv_yLoc}] [-relative_distance_to_sink sinkWeight]
[-off_load_at_location {x1a y1a x1b y1b x2a y2a x2b y2b ....} | { xLoc yLoc inv_xLoc inv_yLoc}]}
[-radius radius ] ]
[-location {xLoc yLoc} | { xLoc yLoc inv_xLoc inv_yLoc} [-buffer_orient {R0 | R90 | R180 | R270 |
MX | MX90 | MY | MY90}]]
[-net netName [-spread_distance distance]]
[-cells {list of cells} ]
[-relative_distance_to_sink sinkWeight | -off_load_at_location {x1a y1a x1b y1b x2a y2a x2b y2b
....} | { xLoc yLoc inv_xLoc inv_yLoc} | -off_load_slack slack | -spread_distance distance | -
spread_count number ]
[-location {xLoc yLoc} | { xLoc yLoc inv_xLoc inv_yLoc} | -no_place]
```

Adds either a single buffer or two inverters on a net. This allows you to optionally specify the location and radius, the location relative to the sink pin, and the buffer insertion based on the slack. This command also checks for and observes power domains based on the specified instance name and location. This command cuts wires by default.

This command does not modify `dont_touch` nets if `-eco_honor_dont_touch` is set to `true`.

Notes:

- The `eco_add_repeater` command is simultaneous setup/hold and MMMC aware. To make the command simultaneous setup/hold aware, set the global variable `set_global timing_enable_simultaneous_setup_hold_mode` to `true`. The simultaneous setup/hold mode can only be used for timing analysis. It cannot be used for any physical implementation steps.
  By default, if you specify an inverter as the master cell, `eco_add_repeater` adds a pair of inverters.
- `eco_add_repeater` can return new instance and net name: `newInstName inputNet outputNet`
  Example
  ```
  set result [eco_add_repeater -net s2_hresp0 -cells inv_32 -pins \
  arm926p_test/wrapper_1_out_reg/d -location 1069.018666 128.2435714]
  ```

```
set newInstName [lindex $result 0]
set inputNetName [lindex $result 1]
set outputNetName [lindex $result 2]
```

If a pair of inverter is added then the return list could be the following, assuming `inv1OutputNet` drives the input pin of `inv2InstName`:

```
{inv1InstName inv1InputNet inv1OutputNet inv2InstName inv2OutputNet}
```

- When `eco_add_repeater` is used in the post-mask ECO, ensure that it is used with the `-no_place` option to avoid base layer change. Also, call `place_eco -use_spare_cells` or `eco_swap_spare_cell` to map the new repeater to the spare cell. If you plan to call `route_eco -route_only_layers bottomLayer:topLayer` later, ensure that the added repeater can be connected through the specified layer. Otherwise, please do not use the `eco_add_repeater` command. You can also use `read_eco <eco_files> -post_mask` to add repeaters in the post-mask ECO.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `eco_add_repeater` parameter. For a detailed description of the command and all of its parameters, use the man command: `man eco_add_repeater`
`-buffer_orient` *orientation*	Specifies the legalized orientation for the instance. Valid orientation values are: R0, R90, R180, R270, MX, MX90, MY, MY90. *Data_type*: string, optional For more information on orientation keys, see "Orientation Key" in the *Floorplanning the Design* chapter of the *Innovus User Guide*.
`-cells` *listOfCells*	Specifies the master name for the cell to be inserted. Either a buffer or an inverter can be specified as the master cell. If an inverter is specified, two inverters in series will be added. The cell should have timing information in the library. *Data_type*: string, optional
`-hinst_guide` *hinstName*	Indicates the module hierarchy in which the `inst` is inserted by Tempus or a third party tool so that Innovus is able to make the association in case the instance appears in the `-pinsList` option of an ECO command at a later stage. This option can be only used when `set_db eco_batch_mode true` is specified. You also need to specify the `-name` option along with it. *Data_type*: string, optional

`-location {xLoc yLoc} | { xLoc yLoc inv_xLoc inv_yLoc}`	Specifies the x and y coordinates for the location of the buffer or inverter pair to insert. If no location is specified, the tool will automatically determine a location. • If the cell is a buffer and location is specified, the newly created buffer is placed at the specified location (x, y). • If the cell is an inverter pair and two locations are specified, the first inverter will be placed at location (x1, y1) and the second inverter will be placed at location (x2, y2). • If the cell is an inverter pair and only one location is specified, both inverters will be placed at the same location (x, y). *Data_type*: (point)+, optional
`-logical_change_only`	Specifies that buffer/inverter addition should be done in the logical-only mode. *Data_type*: bool, optional
`-name` *instName*	Specifies the base name for the repeater to be inserted. *Data_type*: string, optional
`-net` *netName*	Specifies the name of the net where the buffer will be inserted. *Data_type*: string, optional
`-new_net_name` *netName*	Specifies the base name for the new net created for the repeater insertion. *Data_type*: string, optional
`-no_place`	Specifies that the inserted cells should not be placed. Only the logical change in connectivity will be made. *Data_type*: bool, optional
`-off_load_at_location {x1a y1a x1b y1b x2a y2a x2b y2b ....}`	Specifies the location(s) at which buffer (or inverter-pair) should be inserted to drive the sinks downstream from the location(s). *Data_type*: (point)+, optional
`-off_load_slack` *slack*	Inserts a buffer to offload the paths with less than the specified slack threshold from the driver of a net with multiple cells. This option ignores the *term* option, if it is provided. This option is used to connect the added buffer to drive only noncritical receivers, selected By Slack. This checks the timing graph for noncritical receivers and offloads those from the critical path, and could improve critical path timing by penalizing noncritical path delays. *Data_type*: float, optional

`-radius` *radius*	Specifies the radius in which the added instances are free to move. If no legal location can be found in the specified radius, the cells would be placed at the specified location resulting in an overlap with other cells. In that case, you should perform legalization. *Data_type*: float, optional
`-relative_distance_to_sink` *sinkWeight*	Specifies the location of the buffer based on its distance from the sink or the driver pin. The value is a number between 0 and 1. A low value (0.1) places the buffer near the sink; a high value (0.9) places the buffer near the driver. The fraction is based on the length of the wire. *Data_type*: float, optional *Minimum*: 0 *Maximum*: 1 This option works: • When one `term` is provided; it does not work if no `term` or multiple terms are specified. • When `-net` with 1-sink net is provided. It does not work if the specified net is not a 1-sink net.
`-pins` *term1 term2..*	Specifies a list of terms. The terms specified in the list should be connected to a common net and the tool adds buffer to the common net. If `-pins` is not specified, all the terms of the net will be buffered. *Data_type*: string, optional
`-spread_count` *number*	Specifies the number of repeaters to be added for buffering. *Minimum*: 1 *Maximum*: INT_MAX *Data_type*: int, optional
`-spread_distance` *distance*	Specifies the distance (starting from the driver) after which repeaters are to be added. *Minimum*: 2.2250738585072e-308 *Maximum*: 1073741.823 *Data_type*: float, optional

Note: You can specify the following ECO attributes:

```
set_db eco_honor_fixed_status false #to modify FIXED instance.

set_db eco_honor_dont_use false #to modify instances which have dontUse attribute on cells.
```

```
set_db eco_honor_dont_touch false #to modify instance/nets which are marked dontUse.
```

Examples

- The following command adds a buffer near the receiver:
  ```
  eco_add_repeater -net wire_I13 -relative_distance_to_sink 0.1 -cells BUFX4
  ```
- The following command adds a buffer near the driver:
  ```
  eco_add_repeater -net wire_I13 -relative_distance_to_sink 0.9 -cells BUFX4
  ```
- The following command adds a buffer in a specified location
  ```
  eco_add_repeater -net wire_I10 -location 51.5 85.67 -cells BUFFX4
  ```
- The following command adds an inverter by specifying the terminal list:
  ```
  eco_add_repeater -net n1 -cells INVX4 -pins {i1/i2/in1 i1/i2/in2 i1/i2/in3}
  ```
- The following command adds an inverter by specifying the slack time:
  ```
  eco_add_repeater -net wire1_B2 -cells INVX4 -off_load_slack -0.1
  ```

Related Information

- `eco_delete_repeater`
- `place_eco`
- `read_eco`
- `route_eco`
- `timing_enable_simultaneous_setup_hold_mode`
- Floorplanning the Design chapter of the *Innovus User Guide*.

eco_compare_netlist

```
eco_compare_netlist
[-help]
[-logical]
-out_file fileName
[-output_physical_inst_in_file physicalInstFile]
{-def_file referenceFileName [-reference_data {in_memory | external}]}
```

Compares a specified netlist with the design in the database for structural or logical equivalence, and creates a file containing the differences found.

If the `eco_disable_change_net_name_for_flat_netlist` attribute of the `eco` category is set to `true`, there would not be any net renaming for flat netlist. Net renaming will take place if the net name is hierarchical and the terms it connects to do not fall under the same hierarchy or lies up in the hierarchy. Do `set eco_disable_change_net_name_for_flat_netlist 1` before running the `eco_compare_netlist` command to disable net renaming for flat netlist.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `eco_compare_netlist` parameter. For a detailed description of the command and all of its parameters, use the man command: `man eco_compare_netlist`
`-def_file` *`referenceFileName`*	Specifies the name of the DEF file that Innovus uses to compare with the design. *Data_type*: string, required
`-reference_data [in_memory | external]`	Specifies the data from where the post-ECO netlist is to be compared. • `in_memory` compares the post-ECO netlist to the netlist in memory. `in_memory` is the default behavior. • `external` compares the post-ECO netlist to an external netlist. *Data_type*: enum, optional
`-logical`	Compares the netlists for logical equivalence. *Data_type*: bool, optional
`-out_file` *`fileName`*	Specifies the file that contains the differences between the specified DEF file and the design in the database. If the directory specified to save the file does not exist, the tool creates the directory. For example, in the following command if `a/` does not exist, the tool will create it: `eco_compare_netlist -def_file test.def -out_file ./a/test.rpt` *Data_type*: string, required
`-output_physical_inst_in_file` *`physicalInstFile`*	
	When this option is specified, physical instance ECO command is also written to ECO file, and the added physical instances names one per line are dumped into the *`physicalInstFile`* file. *Data_type*: string, optional

Example

- The following command compares the `myFile.def` file against the design, and writes the results to `differences.file`:

  ```
  eco_compare_netlist -def_file myFile.def -out_file differences.file
  ```

Related Information

- eco Category Attributes

eco_delete_repeater

```
eco_delete_repeater
[-help]
[-logical_change_only]
{-insts {list_of_instances} | -inverter_pair {pairs_of_inverters} }
```

Deletes a buffer or pair of inverters connected back-to-back. This command merges wires after ECO.

This command does not modify `dont_touch` nets if `set_db eco_honor_dont_touch` is set to `true`.

Note:

- The `eco_delete_repeater` command is simultaneous setup/hold and MMMC aware. To make the command simultaneous setup/hold aware, set the `timing_enable_simultaneous_setup_hold_mode` attribute to `true`. The simultaneous setup/hold mode can only be used for timing analysis. It cannot be used for any physical implementation steps.
- The `eco_delete_repeater` command cannot be used in the post-mask ECO. Please use `read_eco <eco_files> -post_mask` to delete repeaters in the post-mask ECO.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `eco_delete_repeater` parameter. For a detailed description of the command and all of its parameters, use the man command: `man eco_delete_repeater`
`-insts` *`list_of_instances`*	Specifies a buffer or inverter to delete. If you specify an inverter, the command finds another inverter on the net and deletes the back-to-back pair in order to preserve the logical equivalence. Instances are deleted only if they are tied to each other. To specify a pair of instances, enclose the instances in braces. `-insts` {*`inst1 inst2`*} *Data_type*: string, optional
`-inverter_pair` {*`pairs_of_inverters`*}	Specifies pairs of inverters to be deleted/evaluated for deletion. *Data_type*: string, optional
`-logical_change_only`	Specifies that buffer/inverter deletion should be done in the logical-only mode. *Data_type*: bool, optional

Note: You can specify the following using the `set_db` command:

- `set_db eco_honor_dont_use false` to modify instances which have dontUse attribute on cells.
- `set_db eco_honor_dont_touch false` to modify instance/nets which are marked dontTouch.

Examples

- The following command deletes the specified buffer, `A/inst1`:
  ```
  eco_delete_repeater -insts A/inst1
  ```
- The following command deletes a back-to-back inverter pair containing `A/inv1`:
  ```
  eco_delete_repeater -insts A/inv1
  ```
- The following command deletes instances `DTMF_INST/RESULTS_CONV_INST/FE_PHC845_r1644_2_` and `DTMF_INST/RESULTS_CONV_INST/FE_PHC845_r1643_3_`:
  ```
  eco_delete_repeater -insts {DTMF_INST/RESULTS_CONV_INST/FE_PHC845_r1644_2_
  DTMF_INST/RESULTS_CONV_INST/FE_PHC845_r1643_3_}
  ```

Related Information

- `eco Category Attributes`
- `eco_add_repeater`
- `read_eco`
- `set_db`

eco_design

```
eco_design
[-help]
sessionDirectoroy
{design name}
net_list_file
[-allow_to_load_binary_timing_file]
[-add_hport_for_tie_off]
[-cpf_file newCPFFile | -1801_file new1801File]
[-def defFile]
[-eco_mmmc newMMMCFile]
[-eco_sdc newSDCFile]
[-keep_inst_location]
[-lef_files LEFFilesName]
[-no_add_filler]
[-filler_prefix prefix | -no_delete_filler]
[-route_only_layers MLb:MLt]
[-no_eco_place]
[-no_eco_route]
[-no_timing]
[-post_mask]
[-report_file reportFile]
[-skip_ccopt]
[-spare_cells spareCellName]
[-suffix suffix]
[-tie_cells cellNameList]
[-use_gate_array_cells GACoreSite | -use_gate_array_filler_cells list_of_filler_cells]
```

Takes an Innovus database and a modified netlist as input and performs ECO operations. It restores the {design name}, examines the changes in the new netlist, and automatically implements the required changes with `place_eco` and `route_eco`.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `eco_design`. For a detailed description of the command and all of its parameters, use the man command: man `eco_design`
`-1801_file new1801File`	Specifies the name of the new CPF file. *Data_type*: string, optional

`-add_hport_for_tie_off`	Allows the added tie cells to connect to the tie pins across hierarchical boundaries if other constraints, such as maximum fanout and maximum distance, allow it. This parameter can be used only with `-post_mask` and `-tie_cells` options. *Data_type*: bool, optional
`-allow_to_load_binary_timing_file`	Allows the tool to load the binary timing graph or the timing constraint file. If you execute `eco_design` based on binary timing graph or timing constraints, the tool will give an error and stop loading the new design. This is because the timing graph or the timing constraints file from the loaded design is a binary file, and if you keep on loading it, it may cause fatal error. In such a case, you need to restore the original design and then save a new design with the no-binary timing file. Then, execute `eco_design` based on the new design. If you still want to load the current design, you need to run `eco_design -allow_to_load_binary_timing_file 1` to turn on the control.
`-cpf_file` *newCPFFile*	Specifies the name of the CPF file. *Data_type*: string, optional
`-def` *fileName*	Specifies the DEF file you want the software to load automatically when performing ECO. This parameter automatically assigns the location of the new ECO cells. As a result, you do not need to run `place_eco` after performing the ECO. *Data_type*: string, optional
{*design name*}	Specifies the top cell name of the {design name}. *Data_type*: string, required
`-eco_mmmc` *newMMMCFile*	Specifies the name of the ECO view definition file. *Data_type*: string, optional
`-eco_sdc` *newSDCFile*	Specifies the name of the new SDC file. *Data_type*: string, optional

`-filler_prefix` *prefix*	Specifies the filler cell instance prefix. The `eco_design` command removes all filler cells before placing the ECOs. If you do not specify a prefix, `deleteFiller` removes filler instances based on the `add_fillers` category settings. If you have not set `add_fillers` category attributes, then no filler cells are removed. *Data_type*: string, optional
`-keep_inst_location`	Keeps the instance location of changed cells. *Data_type*: bool, optional
`-lef_files` *LEFFilesName*	Specifies the LEF files name for loading. Use this option if you want to replace the existing LEF files in the original design with new LEF files. Refer to `read_db -lef_files` for more details and limitations. For example, tech.lef, A.lef are the existing LEFs used in original design. You can change A.lef to A_v2.lef using the following command: `eco_design original.enc.dat myDesign myDesign.new.v -lef_files {tech.lef A_v2.lef}` *Data_type*: string, optional
`-no_add_filler`	Skips filler insertion in the pre-mask eco flow. *Data_type*: bool, optional
`-route_only_layers` *MLb:MLt*	Specifies the layers to be modified during the ECO process. *Data_type*: string, optional
net_list_file	Specifies the name of new netlist file. *Data_type*: string, required
`-no_delete_filler`	Specifies to not remove physical cell instances, such as filler cells, end-cap cells, and well-tap cells. *Data_type*: bool, optional
`-no_eco_place`	Prevents placement during the ECO process. *Data_type*: bool, optional

`-no_eco_route`	Prevents routing during the ECO process. *Data_type*: bool, optional
`-no_timing`	Ignores timing libraries and MMMC configuration.
`-post_mask`	Specifies that the changes are being made on a post-mask {design name}. `-post_mask` is required when the `-suffix`, `-spare_cells`, `-modifyOnlyLayers`, `-tie_cells`, or `-add_hport_for_tie_off` option is specified. *Data_type*: bool, optional
`-report_file` *reportFile*	Specifies the name of the report file. *Data_type*: string, optional **Note**: By default, `eco_design` creates the default `ecoRestoreDesign.rpt` file. However, this default file will be overwritten by the file name specified using the `-report_file` option in next eco RUN.
session_directory	Specifies the .dat directory from which to load the files. *Data_type*: string, required
`-skip_ccopt`	Skips restore ccopt config for `time_design_signoff`.
`-spare_cells` *spareCellName*	Specifies the name of the spare cell to use during the ECO process. *Data_type*: string, optional
`-suffix` *suffix*	Specifies the suffix to be appended to the spare cells. You can specify this parameter only if you specify `-post_mask`. *Default:* `_SPARE` *Data_type*: string, optional

`-tie_cells "cellName1 cellName2"`	Specifies the existing tie cells to be reused to make tie connections for newly created SPARE instance in the {design name}. You can only specify a maximum of two tie-cells, where one cell must be a tie-high driver, and the other a tie-low driver. The two tie-cell names must be surrounded by quotation marks. This parameter can be used only with `-post_mask` option. *Data_type*: string, optional
`-use_gate_array_cells GA_site`	Specifies the site name of the GA cell. Check the site name defined in the LEF file. *Data_type*: string, optional
`-use_gate_array_filler_cells list_of_filler_cells`	Specifies a list of Gate Array (GA)-style filler cells that the tool can replace with GA cells during post-mask ECO. You can only use this parameter in a post-mask ECO flow. *Data_type*: string, optional **Note:** You cannot specify the `-use_gate_array_cells` parameter if you use this parameter.

Examples

- The following command performs a pre-mask ECO:
  ```
  eco_design original.enc.dat myDesign myDesign.new.v
  ```
- The following command and options are used to implement a post-mask ECO:
  ```
  eco_design -post_mask -route_only_layers 2:3 -spare_cells *spare* original.enc.dat myDesign
  myDesign.new.v
  ```

Related Information

- `route_eco`
- `place_eco`
- `read_db`
- ECO Flows chapter in the *Innovus User Guide*

eco_oa_design

```
eco_oa_design
[-help]
lib
cell
view
-eco_verilog_file input_fileName
[-post_mask [-suffix suffixName]]
[-use_gate_array_cells GACoreSite]
[-report_file output_fileName]
```

Imports an OpenAccess database and an ECO-modified netlist. Restores the physical objects in the database, examines the changes in the new netlist, and implements the required connectivity changes in the logical database without making changes to the physical objects. It preseves the spare cell markings in the old `lib`/`cell`/`view`.

For more information, refer to the Innovus-Based ECO Flow section of the `Chip Finishing and ECO Flows` chapter of *Innovus Stylus Common UI Mixed Signal (MS) Interoperability Guide*.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `eco_oa_design` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man eco_oa_design`.
`cell`	Specifies the OpenAccess cell name of the top cell to import. *Data_type*: string, required
`lib`	Specifies the OpenAccess library name. *Data_type*: string, required
`-eco_verilog_file` *`input_fileName`*	Specifies the changed Verilog netlist. *Data_type*: string, required **Note:** The original netlist must be written out from the original *`lib cell view`* and it must be hand-modified to incorporate the required ECO changes.

`-report_file` *output_fileName*	Specifies a report file that shows the additions and removals of instances and nets in the new Verilog netlist. *Default:* eco.log *Data_type*: string, optional
`-post_mask`	Specifies that eco_oa_design works in the post-mask mode. If this parameter is not specified, then Innovus considers the mode as pre-mask by default. By default, the suffix is _SPARE. *Data_type*: bool, optional
`-suffix` *suffixName*	Specifies the suffix name for the spare cells. *Default*: _SPARE *Data_type*: string, optional
`-use_gate_array_cells` *GACoreSite*	The `-use_gate_array_cells` parameter is valid only in the post-mask mode. *Data_type*: string, optional 1. For cells that match the specified GA core site: ◦ Newly added GA cells that exist in memory but not in the OpenAccess database are left unplaced. ◦ GA cells that exist in the OpenAccess database but not in the memory are deleted in the new database. 2. For cells that do not match the specified GA core site, Innovus treats them as standard cells as described in post-mask mode.
view	Specifies the OpenAccess view name to import. *Data_type*: string, required

Examples

The following command reads the modified Verilog from the `eco.v` file, complete physical pre-ECO database from the `lib/cell/view designLib/TOP/final` in the post-mask mode:

```
eco_oa_design designLib TOP final -eco_verilog_file eco.v -post_mask
```

In post-mask mode, no physical database changes are allowed and the new logical changes coming through `eco.v` are incorporated using spare cells in successive `place_eco` and `eco_route` commands.

Related Information

- Chip Finishing and ECO Flows chapter of *Innovus Stylus Common UI Mixed Signal (MS) Interoperability Guide*.

eco_read_def

```
eco_read_def
[-help]
[-post_mask [-suffix suffixName]]
[-use_gate_array_cells GACoreSite]
[-report_file in_file]
[-drc_clean_data]
[-keep_inst_location]
[-merge_floating_route_to_pg_net netName]
[-eco_def_file {new_DEF_in_file}]
filename
```

Restores physical information from an old design and compares this information with the design in memory. This command uses the specified DEF file in the following ways for pre-mask and post-mask ECO:

- Used in pre-mask mode (without the `-post_mask` parameter) this command behaves as follows:
    - New cells and nets
      Cells that exist in memory but not in the DEF file become unplaced.
    - Deleted cells or nets
      Cells and nets that exist in the DEF file but not in memory are discarded.
    - Modified nets
      Nets found both in the DEF file and in memory, but whose connections are different, are marked as "modified" for further processing during `route_eco`.
    - Matched cells/nets
      Nets and cells that exist both in the DEF file and in memory, and have the same connections, are placed as described in the DEF file. This includes the soft matching capability used in `read_def`.

Soft matching happens when a DEF net name does not have an equivalent name and another net is found in memory, that has the same connections as described in the DEF file.

The most common case for soft matching is when nets have multiple aliases in a hierarchical design `"net1" = "inst1/net2" = "inst1/inst2/net3"` and so on. Any of these net names can be used in the DEF and can have the same connections.

Without soft matching, net `a', for example, is removed and net `b' is created in it's place, resulting in ripping of the wire.

- In post-mask mode (with the -post_mask parameter), the software can only change nets, not cells. This command behaves as follows:
    - New added cells and nets
      Cells that exist in memory but not in the DEF file become unplaced. Later, these cells will be mapped to spare cells when you use `place_eco -use_spare_cells`.

- Deleted cells
  Cells and instances that exist in DEF but not in memory are added to a spare cell list in Innovus: They are not discarded. Deleted cells are renamed with their original name and the suffix you specify. The default suffix is `_SPARE`. For example, `cellA` is renamed `cellA_SPARE`. The software retains deleted nets and their routing, which `route_eco` reroutes later.
- Deleted nets
  Nets that exist in DEF but not in memory are kept and processed later when you use the `route_eco` command.
- Modified nets
  Same behavior as in the pre-mask mode.
- Matched cells and nets
  Same behavior as in the pre-mask mode.
- Physical cells
  Cells that exist in the DEF file but not in memory, which are marked with `+SOURCE DIST` in the DEF file, are restored.

If you specify `-use_gate_array_cells`, this implies a post-mask mode. This option behaves as follows:

- For cells that match the specified GA core site:
  - New added GA cell that exists in memory but not in the DEF file are left unplaced.
  - New GA cells that exist in the DEF file but not in memory are deleted.
- For cells that do not match the specified GA core site, the software treats them as standard cells as described in `-post_mask` mode.

If you specify a report filename, this command reports the following information:

- New cells and nets found in memory but not in the DEF file
- Deleted cells and nets found in DEF but not in memory
- Nets whose connection does not match
- Cells and nets that match in Innovus and DEF

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `eco_read_def` parameter. For a detailed description of the command and all of its parameters, use the man command: `man eco_read_def`

`-drc_clean_data`	Specifies that the routed data is DRC clean. ECO routing will skip routing analysis. *Data_type*: bool, optional
`-eco_def_file {`*`new_def_in_file`*`}`	Specifies the name of a partial DEF file. The tool compares the original routed DEF file or Innovus database with this file and marks the changed instances, changed location of the existing instances, and changed segments or wires. *Data_type*: string, optional
`in_file`	Specifies the name of the routed/original DEF file to load. *Data_type*: string, required
`-keep_inst_location`	Applies the placement status and location of the instance in the DEF file to the instance in db. *Data_type*: bool, optional This option is only applicable in the following special situation: When an instance in DEF file and an instance in db have the same name but the master cell of the instance in the DEF and in the db do not match, then the instance in DEF is considered as deleted while the instance in db is added. In this case, if the option is not specified, the general rule for addition or deletion of an instance is applied as described earlier. For the added instance, the placement status is set to unplaced. If the option is specified, the placement status and location of the instance in the DEF file is applied to the instance in db.
`-merge_floating_route_to_pg_net` *`netName`*	Merges the routing data to the specified PG net. In this case, the physical-only net will not be created. When this parameter is not specified, if a net is deleted in the new netlist but exists in the DEF file, then the `eco_read_def` command discards the net in pre-mask ECO and creates a physical-only net in post-mask ECO to preserve the routing data on the net. *Data_type*: string, optional
`-post_mask`	Runs this command in post-route mode. *Data_type*: bool, optional
`-report_file` *`in_file`*	Specifies the name for the report generated by the `eco_read_def` command. *Default:* `eco_read_def.rpt` *Data_type*: string, optional

`-suffix` *suffixName*	Appends the specified suffix to cells that appear in the DEF file but have been deleted in the new netlist. *Default:* `_SPARE` *Data_type*: string, optional
`-use_gate_array_cells` *GACoreSite*	Specifies the GA Core Site. *Data_type*: string, optional

Examples

The following example reads a DEF file, `myOld.def`, for post-mask ECO.

```
eco_read_def -post_mask -suffix _SPARE -report_file eco_read_def.rpt myOld.def
```

When the software converts a deleted instance to a spare cell, it adds the `_SPARE` suffix to the instance. This command generates a report `eco_read_def.rpt`.

The following example shows a report file:

```
DELINST DTMF_INST/TDSP_CORE_INST/PROG_BUS_MACH_INST/i_9649 INVX1
(converted to spare cell
DTMF_INST/TDSP_CORE_INST/PROG_BUS_MACH_INST/i_9649_SPARE)

ADDINST DTMF_INST/TDSP_CORE_INST/PROG_BUS_MACH_INST/i_9649 BUFX8
MODIFYNET dtmf_chip_TIELO_Y_r2c1
ADDNET DTMF_INST/UNCONNECTED_000
ADDNET DTMF_INST/UNCONNECTED_001
ADDNET DTMF_INST/TDSP_CORE_INST/UNCONNECTED_000
ADDNET DTMF_INST/TDSP_CORE_INST/UNCONNECTED_009
ADDNET DTMF_INST/TDSP_CORE_INST/UNCONNECTED_010
```

Related Information

- place_eco
- `read_def`
- `route_eco`

eco_remove_tied_inputs

```
eco_remove_tied_inputs
[-help]
[-base_cells libCell/ptnCell+]
[-ignore_base_cells libCell/ptnCell+]
[-ignore_insts inst+]
[-insts inst+]
[-relaxed_placement]
[-term term_list]
[-verbose]
[-report]
```

Remaps or restructures gates that have inputs connected to the same net. These gates are remapped into functionally equivalent logic such that none of the new gates has inputs tied together. Usually, such optimization is good in all objectives. For example, a 4-input OR gate with 3 of its inputs tied together (implementing function a+b+b+b) can safely be replaced with a 2-input OR gate (implementing a+b). This move is likely to be better in terms of area, power, and delay. However, in case of complex gates, it may not be possible to restructure a gate in a way that improves delay, but it may be possible to decompose it into smaller discrete gates, none of which has inputs tied together. Area and/or delay may be worse, but in some libraries, it is preferable to not have instances with inputs tied together. The `eco_remove_tied_inputs` command fixes any gate with two or more inputs connected to the same net regardless of the improvement in delay/area/power. It tries to find improving moves, but it may accept moves that worsen delay/area/power.

The command can also insert buffers in nets tying inputs together. Inserting buffers should be used for the gates that cannot be automatically fixed or that are preserved. You need to specify which buffer cell(s) must be inserted before which input(s). There must be as many buffer cells specified as there are inputs specified. The only exception is specifying a single buffer cell; in that case it is understood that all inputs will be untied by using instances of that same buffer cell.

The command honors `opt` category settings for timing update and legalization. It does not call `route_eco` if run in the postRoute/ECO stages.

Note: MBFF cells are ignored by default as they always have tied inputs.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `eco_remove_tied_inputs` parameter. For a detailed description of the command and all of its parameters, use the man command: `man eco_remove_tied_inputs`
`-base_cells libCell/ptnCell+`	Specifies the buffer cells to be used for fixing. *Data_type*: (base_cell)+, optional
`-ignore_base_cells libCell/ptnCell+`	Specifies to ignore instances of these cells from fixing. *Data_type*: (base_cell)+, optional

`-ignore_insts` *inst+*	Specifies to ignore these instances from fixing. *Data_type*: (inst)+, optional
`-insts` *inst+*	Specifies to fix only these instances. *Default*: All design minus ignored instances/cells. *Data_type*: (inst)+, optional
`-relaxed_placement`	Accepts moves even if instances cannot be placed. *Data_type*: bool, optional You do not need to set this option if `eco_remove_tied_inputs` can successfully fix all violations. In some cases, for example if an instance with tied inputs is in a very congested area, the tool may not be able to place the new instances. In such cases, you can use the `-relaxed_placement` option to fix such instances. However, the final placement may not be legal, and the new instances will likely overlap. For this, you can perform incremental placement to legalize the placement.
`-term` *term_list*	Specifies the terms to be buffered. *Data_type*: string, optional
`-verbose`	Reports existing tied inputs, but does not provide fixing details by any method. *Data_type*: bool, optional
`-report`	Specifies to increase the detail of reports and fixing. *Data_type*: bool, required

Related Information

- `opt Category Attributes`
- `route_eco`

eco_swap_spare_cell

```
  eco_swap_spare_cell
[-help]
instName
spareInstName
[-keep_scan]
[-no_scan_trace]
[-preserve_pins {list_of_spare_cell_pins}]
[-suffix suffix]
```

Swaps an unplaced or placed cell with a cell in the spare cell list.

You can specify this command only after you have run `set_spare_insts`.

This command can swap cells of the same cell type (same footprint and identical pin name). Cells of different sizes can also be swapped. The spare cell is connected according to the specified global net connections.

If the cell to be swapped is unplaced, it is mapped to the spare cell. *instName* is deleted, and its connection is transferred to the spare cell. If the cell to be swapped is placed, it is swapped with the spare cell and is renamed to *instName suffix* if the `-suffix` option is used. If a suffix is not specified, the *inst name* cell is renamed to *spareInstName*. The *instName* cell's connections are transferred to *spare instName*. The input of *instName* is `tielo`, based on the global connection definition.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `eco_swap_spare_cell` parameter. For a detailed description of the command and all of its parameters, use the man command: `man eco_swap_spare_cell`
`-keep_scan`	Keeps the scan connection for both the spare cell and the specified instances. Before you can use this parameter, you must first identify the scan connections by running a scan trace. *Data_type*: bool, optional
`instName`	Specifies the name of the cell to swap. *Data_type*: (inst), required
`-no_scan_trace`	Disables scan tracing while swapping. *Data_type*: bool, optional
`-preserve_pins {`*`list_of_spare_cell_pins`*`}`	Specifies a list of pins whose connection you want to preserve while swapping spare cells. The same pin on both the spare cell and specified instances is preserved. If both the instance and the spare cell are placed instances, the `-preserve_pins` parameter is effective for both cells. If the instance is unplaced, `-preserve_pins` only affects the spare cell because the instance is deleted. *Data_type*: string, optional
`spareInstName`	Specifies the spare cell in the spare cell list to swap with the placed cell. *Data_type*: (inst), required
`-suffix` *`suffix`*	Specifies a string that is appended to the *`instName`* after swapping. *Data_type*: string, optional

Examples

The following example swaps a placed cell, `DTMF_INST/TDSP_CORE_INST/PROG_BUS_MACH_INST/i_9649`, with spare cell `spare1`:

```
set_spare_insts -inst *spare*
place_eco -use_spare_cells
check_place

eco_swap_spare_cell DTMF_INST/TDSP_CORE_INST/PROG_BUS_MACH_INST/i_9649 spare1
```

Related Information

- check_place
- set_spare_insts
- place_eco

eco_update_cell

```
eco_update_cell
[-help]
-cells list_of_cells |-up_size | -down_size
[-orient {r0 | r90 | r180 | r270 | mx | mx90 | my | my90}]
-insts instNames
[-location {xLoc yLoc}]
[-pin_maps oldpin1 newpin1 oldpin2 newpin2 ...]
```

Upsizes or downsizes the specified instance based on the available footprint. Cell mapping is also possible as long as the output pins are functionally the same. This command also checks for and observes power domains.

This command does not modify `dont_touch` nets if the `eco` attribute `eco_honor_dont_touch` is set to `true`.

Notes:

- If you use the `eco_update_cell` command to modify a `` `fixed' `` instance, the software issues a warning message.
  Instead, you can specify the following using the `eco` category attributes:
  `eco_honor_fixed_status` to `false`, to modify FIXED instance.
  `eco_honor_dont_use` to `false` to modify instances which have dontUse attribute on cells.
  `eco_honor_dont_touch` to `false` to modify instance/nets which are marked dontTouch.
- `The eco_update_cell` command is simultaneous setup/hold and MMMC aware. To make the command simultaneous setup/hold aware, set the `timing_enable_simultaneous_setup_hold_mode` attribute to `true`. The simultaneous setup/hold mode can only be used for timing analysis. It cannot be used for any physical implementation steps.
- When resizing instances inside multiple-instantiated blocks, the resize would be applied to all instantiations of the block.
- You can change a buffer to a pair of inverters using the `eco_delete_repeater` and `eco_add_repeater` commands. You can change a buffer to an inverter by first using the `eco` attribute `eco_check_logical_equivalence false` and then using `eco_update_cell`. However, please note that it results in the change in logic.
- The `eco_update_cell` command cannot be used in the post-mask ECO. Please use `read_eco <eco_files> -post_mask` to change cells in post-mask ECO.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `eco_update_cell` parameter. For a detailed description of the command and all of its parameters, use the man command: `man eco_update_cell`
`-orient {r0 | r90 | r180 | r270 | mx | mx90 | my | my90}`	Specifies the legalized orientation for the instance. You can specify this parameter with the `-location` parameter only. *Data_type*: string, optional
`-insts` *instNames*	Specifies the names of the instances to be added. *Data_type*: string, optional
`-location {xLoc yLoc}`	Specifies the legalized location where the instance will be placed. *Data_type*: point, optional
`-up_size`	Upsizes the specified instance with the next available cell in the same footprint unless the specified cell is already of the highest driving strength. Upsizing an instance that drives a large load can improve the driver delay and the transition time at the receivers. *Data_type*: bool, optional
`-down_size`	Downsizes the specified instance with the next available cell in the same footprint unless the specified cell is already of the lowest driving strength. Downsizing an instance on the noncritical path can reduce the loading of its driver on the critical path. *Data_type*: bool, optional
`-cells` *list_of_cells*	Specifies the name of a specific cell(s) to be substituted in the design. The function of the original cell has to match the function of the new cell for all the outputs used by the old cell. The original master cell and the new master cell must have exactly same number of pins. *Data_type*: string, optional
`-pin_maps` *oldpin1 newpin1 oldpin2 newpin2*	Specifies the pin mapping for the new cell based on the old cell. This option is required if the new master cell has different pin names than the original cell. The net connected to *oldpin1* of the original cell should be connected to *newpin1* of the new cell, and so forth. *Data_type*: string, optional

Examples

- Upsizes the specified instance:
  ```
  eco_update_cell -insts Top_I18 -up_size
  ```

- Downsizes the specified instance:

  ```
  eco_update_cell -insts Top_I18 -down_size
  ```

- Modifies the specified instance according to the specified cell:

  ```
  eco_update_cell -insts Top_I18 -cells BUFX20
  ```

- Changes to another cell with a different pin name:

  ```
  eco_update_cell -insts Top_I27 -cells TXOR2X2 -pin_maps A X B Y Y Z
  ```

- Downsizes the specified instance:

  ```
  eco_update_cell -insts Top_I18 -down_size
  ```

Related Information

- eco_add_repeater
- eco_delete_repeater
- eco Category Attributes
- read_eco

eco Category Attributes

The root attributes in the eco category are:

- eco_add_ports
- eco_batch_mode
- eco_delay_calculation_effort
- eco_disable_change_net_name_for_flat_netlist
- eco_honor_dont_touch
- eco_honor_dont_use
- eco_honor_fixed_status
- eco_honor_fixed_wires
- eco_honor_power_intent
- eco_inherit_net_attribute
- eco_check_logical_equivalence
- eco_prefix
- eco_preserve_hpin_function
- eco_refine_place
- eco_spread_inverter
- eco_update_only_layers
- eco_update_timing

To set the value of any root attribute, use the `set_db` command:

```
set_db attribute_name value
```

To get the current value of any root attribute, use the `get_db` command:

```
get_db attribute_name
```

To get the current value of all attributes in the `eco` category, use the following command:

```
get_db -category eco
```

eco_add_ports

`eco_add_ports {true|false}`

Default: `true`

Data_type: bool, read/write

Controls whether timing optimization adds ports to the lower-level hierarchies of the design. It only works during ECO and overwrites the value of `opt_add_ports` for ECO commands, `eco_add_repeater`, `eco_update_cell`, and `eco_delete_repeater`.

eco_batch_mode

```
eco_batch_mode {true|false}
```

Default: `false`

Data_type: bool, read/write

Speeds up the runtime in case the user performs lots of interactive ECO operations. When the user specifies `eco_batch_mode true`, various ECO operations are performed in a batch and initialization steps are not performed for every ECO call. In addition, updating timing is performed only once for all the ECO operations. Tool will only perform timing initialization at the first ECO command in the batch mode and timing update at the time of exiting the batch mode. User needs to set batch mode false to exit the batch mode.

The batch mode is activated by `set_db eco_batch_mode true` and exited by `set_db eco_batch_mode false`, and only `eco_add_repeater`, `eco_delete_repeater` and `eco_update_cell` could be given in between `set_db eco_batch_mode true` and `set_db eco_batch_mode false`:

```
set_db eco_batch_mode true
eco commands (eco_add_repeater, eco_delete_repeater, eco_update_cell)

set_db eco_batch_mode false
```

If a user activates the batch mode, then the user needs to remember to exit it. If the user does not exit the batch mode, the user might see some unexpected results. For example, if the tool automatically added cell padding for blocks when `place_detail` is called by ECO commands for some reason, the cell padding will not be removed and the block size will change unless the user exits the batch mode.

All other settings should be made prior to activating the batch mode. If you would like to change settings, you need to exit the batch mode first.

Note: This option supports all commit ECO commands. Evaluation commands are not supported.

Related Command

- `place_detail`

eco_delay_calculation_effort

```
eco_delay_calculation_effort {low | medium | high}
```

Default: `high`

Data_type: string, read/write

Specifies delay calculation effort level { low | medium | high} used in timing update.

eco_disable_change_net_name_for_flat_netlist

`eco_disable_change_net_name_for_flat_netlist`

Default: `0`

Data_type: int, read/write

Specifies whether nets would be renamed in ECO report of `eco_compare_netlist`. When set to 1, there will not be any net renaming for flat netlist. The net renaming will take place if net name is hierarchical and the terms it connects to do not fall under the same hierarchy or lies up in the hierarchy. You need to set `eco_disable_change_net_name_for_flat_netlist` to `1` before `eco_compare_netlist` command is issued to disable net renaming for flat netlists.

eco_honor_dont_touch

`eco_honor_dont_touch {true|false}`

Default: `true`

Data_type: bool, read/write

Checks for the don't touch attribute on nets and instances. If you specify a terminal list, this parameter checks each terminal for the don't touch attribute. The `eco_add_repeater` command honors the don't touch nets. The `eco_delete_repeater` and `eco_update_cell` commands honor don't touch instances.

Related Commands

- `eco_delete_repeater`
- `eco_update_cell`

eco_honor_dont_use

`eco_honor_dont_use {true|false}`

Default: `true`

Data_type: bool, read/write

Checks for the don't use attribute on cells. The `eco_add_repeater`, `eco_delete_repeater`, and `eco_update_cell` commands honor don't use cells.

Related Commands

- `eco_add_repeater`
- `eco_delete_repeater`
- `eco_update_cell`

eco_honor_fixed_status

`eco_honor_fixed_status {true|false}`

Default: `true`

Data_type: bool, read/write

Checks for fixed instances.

It restricts the resize or deletion of fixed instances.

eco_honor_fixed_wires

`eco_honor_fixed_wires {true|false}`

Default: `true`

Data_type: bool, read/write

Checks for fixed net wires.

It restricts the addition or deletion on fixed net so that the fixed status of the wires is not altered.

When `eco_honor_fixed_wires` is set to `false`, `eco_add_repeater` can insert a cell on the wires with the FIXED/COVER status and change the status of these wires from FIXED/COVER to ROUTED.

eco_honor_power_intent

`eco_honor_power_intent {true|false}`

Default: `true`

Data_type: bool, read/write

Performs MSV checks during ECO. When set to `true`:

- `eco_update_cell` does not allow:
  - Resizing a regular cell with an always-on cell (and vice-versa).
  - Resizing with cells belonging to different power domains.
- `eco_add_repeater` does not allow:

  - Adding buffer to a power domain it does not belong to.
  - Buffering a cross power domain net if it can cause MSV violations.
  - Adding a regular buffer to a net that requires an always-on buffer.
  - Adding an always-on buffer to a net that does not require an always-on buffer.
- `eco_delete_repeater` does not allow deletion of a buffer or an inverter-pair that will cause:
  - A redundant isolation cell.
  - A CPF-constrained net to be deleted.

Note: To prevent buffer insertion failure due to MSV constraints, you need to set `eco_honor_power_intent` to `false`.

eco_inherit_net_attribute

```
eco_inherit_net_attribute {true|false}
```

Default: `true`

Data_type: bool, read/write

Inherits the net attribute.

eco_check_logical_equivalence

```
eco_check_logical_equivalence {true|false}
```

Default: `true`

Data_type: bool, read/write

Enables functionality checking when swapping cells. The software can swap cells that are logically equivalent.

When `eco_check_logical_equivalence` is `true`:

- `eco_add_repeater` adds inverter pair if the specified cell is inverter.
- `eco_delete_repeater` deletes inverter pair if the specified instance is inverter.
- `eco_update_cell` changes cell to another cell type with the same function as long as they have same number of pins.

When `eco_check_logical_equivalence` is set to `false`:

- `eco_add_repeater` can add only one single inverter if the specified cell is inverter.
- `eco_delete_repeater` can delete only one single inverter if the specified instance is inverter.
- `eco_update_cell` can perform nonequivalent cell change as long as they have same number of pins.

Related Commands

- `eco_add_repeater`
- `eco_delete_repeater`
- `eco_update_cell`

eco_prefix

`eco_prefix` *`prefix`*

Default: `ECO`

Data_type: string, read/write

Adds a prefix to the inserted cells.

eco_preserve_hpin_function

`eco_preserve_hpin_function {true|false}`

Default: `false`

Data_type: bool, read/write

Determines whether to preserve logical functions at hierarchical module ports. It only works for manual ECO commands and overwrites the value of `opt_preserve_hpin_function` for the ECO commands, `eco_add_repeater`, `eco_update_cell`, and `eco_delete_repeater`.

eco_refine_place

`eco_refine_place {true|false}`

Default: `true`

Data_type: bool, read/write

Legalizes placement whenever `eco_add_repeater` or `eco_update_cell` are used. If `false`, you must run `place_detail` after ECO if you want to legalize placement.

Note: Specifying `eco_refine_place true` can affect performance when large numbers of cells are added or modified.

Example

The following command does not legalize placement whenever `eco_add_repeater` or `eco_update_cell` are used:

```
set_db -eco_refine_place false
```

Related Commands

- `eco_add_repeater`
- `eco_update_cell`
- `place_detail`

eco_spread_inverter

```
eco_spread_inverter {true|false}
```

Default: `true`

Data_type: bool, read/write

Distributes an inverter pair evenly between the driver and the sink. It works when the buffering nets have only one driver and one sink. The design must be routed before you use this parameter.

eco_update_only_layers

```
eco_update_only_layers bottom_layer:top_layer
```

Default: `""`

Data_type: string, read/write

Modify routes only on metal layers in this range (both inclusive).

eco_update_timing

```
eco_update_timing {true|false}
```

Default: `true`

Data_type: bool, read/write

Controls whether to `eco*` commands recompute timing. If `true`, the eco* commands recompute timing. If `false`, the commands do not recompute timing, and us can later use `report_timing` to recompute timing.

ECO Without Category Attributes

The ECO root attributes without any category are:

- bit_blasted_port_style
- group_instance_suffix

To set the value of any root attribute, use the `set_db` command:

```
set_db attribute_name value
```

To get the current value of any root attribute, use the `get_db` command:

```
get_db attribute_name
```

bit_blasted_port_style

```
bit_blasted_port_style %s_%d
```

Default: `%s_%d, persistent`

Data_type: string, read/write

Defines the bus bit blasted port name.

Specifies the naming style to be used if mapped ports are bit blasted with the command `bitblast_ports`. The values of this global need to contain elements %s and %d.

The following rules apply:

- A port name always starts with the base name (represented by %s)
- The suffix is appended to the base name to form port name, according to the format specified in the global value.
- %d represents the bit information.

Example

The default of %s_%d will convert a bus-bit A[0] to A_0.

If you want to keep the original bus-bit name but have it turned into an escaped scalar name you can use:

```
set_db bit_blasted_port_style {%s\[%d\]}
```

This will convert a bus-bit A[0] to an escaped scalar name A\[0\] in Tcl, so the port .name would be “A[0]”, the .escaped_name would be “A\[0\]”, the dual-ported-object name would look like {port:top/A\[0\]} with Tcl adding the {} around it to protect the \ chars, and the Verilog port would look like “\A[0] “.

One advantage of this is that SDC commands and Innovus Tcl commands that use port names with A[0] will still “soft-match” the new A\[0\] name, so they do not have to change. However, using escaped names may require

careful Tcl scripting to make sure you do not inadvertently lose the escaped \ chars when you want them.

group_instance_suffix

`group_instance_suffix` *`suffix`*

Default: `i`

Data_type: string, read/write

This suffix is added to a new hinst name created by the `group` command. So, `group -name abc` creates a new module named `abc`, and a new `hinst` named `abci`. If you do not want any suffix, you need to set the value to the empty string “”.

end_eco

```
end_eco
[-help]
```

Ends logging ECO operations.

Parameters

None

Examples

- The following commands show the usage of `initECO` and `endECO`.

  ```
  init_eco File1
  ....
  <ECO commands>
  ....
  end_eco
  ```

  All operations performed by ECO Commands are logged in *`File1`*.

fix_multi_drivers

```
fix_multi_drivers
[-help]
[-exclude_net netName | netNameList]
[-replace_bus_bit]
[-report_only]
```

Fixes multi-drive issues by disconnecting driving ports that do not have a real driver.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `fix_multi_drivers` parameter. For a detailed description of the command and all of its parameters, use the man command: `man fix_multi_drivers`
`-exclude_net` *netName* `|` *netNameList*	Specifies the list of nets to be excluded.
`-replace_bus_bit`	Replaces all bus ports with wrong directions, if any, by new scalar ports. If not specified, just report without such replacement. *Data_type*: bool, optional
`-report_only`	Reports, but does not fix, ports with wrong directions. *Data_type*: bool, optional

group

```
group
[-help]
inst|hInst+
[[-module_name module_name -hinst_base_name hinst_base_name ]]
```

Creates a new module and hinst from a group of the specified hinsts or insts. You can only group hinsts and insts that belong to the same hierarchy (the same hinst parent). Returns the newly created hinst.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `group`. For a detailed description of the command and all of its parameters, use the man command: `man group`
inst|hInst+	Specifies the hinst(s) or inst(s), in the same hierarchy, to be grouped. *Date_type*: (inst|hinst)+, required
`-hinst_base_name` *hinst_base_name*	Specifies the hinst base name of the new group instantiation. *Data_type*: string, required
`-module_name` *module_name*	Specifies the module name of the new group. *Data_type*: string, required

Example

The following command groups two hinsts, `h1` and `h2`, at the top-level into a new module named `test`. The new hinst is named testi by default.

```
>group -name test {h1 h2}
```

Related Information

- `group_instance_suffix` attribute in ECO Without Category Attributes
- `ungroup`

gui_show_buffer_tree

```
gui_show_buffer_tree
[-help]
-net netName
[-out_file fileName]
```

Displays a buffer tree associated with the specified net.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `gui_show_buffer_tree` parameter. For a detailed description of the command and all of its parameters, use the man command: `man gui_show_buffer_tree`
`-net netName`	Specifies the name of the net for which you want to display the buffer tree. *Data_type*: string, required
`-out_file fileName`	Specifies the text file containing buffer locations. *Data_type*: string, optional

Example

- The following command highlights instances and wires in the buffer tree associated with the net `reset` in the design display area:
  ```
  gui_show_buffer_tree -net reset
  ```

init_eco

```
init_eco
[-help]
eco_file_name
```

Logs ECO operations in a log file.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `init_eco` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man init_eco`.
`eco_file_name`	Specifies the name of the ECO log file. *Data_type*: string, required

Examples

- The following commands show the usage of `init_eco` and `end_eco`.

  ```
  init_eco File1
  ....
  <ECO commands>
  ....
  end_eco
  ```

  All operations performed by ECO commands are logged in *`File1`*.

Related Information

- `end_eco`

place_eco

```
place_eco
[-help]
[-fix_placed_insts {true|false}]
[-timing_driven {true|false}]
[-use_gate_array_cells GACoreSite]
[-use_gate_array_filler_cells GAFillerCells]
[-use_gate_array_filler_groups {true|false}]
[-use_spare_cells {true|false}]
```

Incrementally places the unplaced standard cells. In a pre-mask flow, this command moves the unplaced standard cells into the core area. During an ECO loop, some new unplaced standard cells may get introduced. If you need to move the unplaced standard cells into the core area and place them without disturbing the original placement (pre-mask flow), it is recommended to run `place_eco` at this stage. In a post-mask flow, if you specify `-use_spare_cells`, the software can map the unplaced cells to the available spare cells. When `-use_gate_array_filler_cells` is provided in a post-mask flow, the command can map existing unplaced cells to placed GAFillers in the design.

You can use this command after the `eco_read_def` command when some newly added cells are still unplaced.

Parameters

`-fix_placed_insts {true | false}`	
	Specifies that instances with the PLACED status are not moved. *Data_type*: bool, optional
`-help`	Outputs a brief description that includes type and default information for each `place_eco` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man place_eco`.
`-timing_driven {true|false}`	Turns on the slack-driven placement for ecoPlace. *Default*: `false` *Data_type*: bool, optional
`-use_gate_array_cells GACoreSite`	

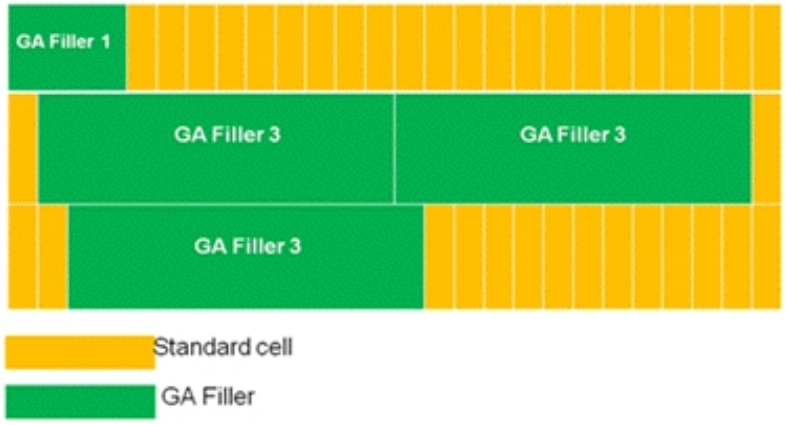
	Selects and places the unplaced instances with techSite matching the provided GACoreSite. It is expected that there is free space to place the instances. If not, GA fillers could be deleted before `place_eco -use_gate_array_cells` and re-inserted after the command. You can use this parameter only for post-mask ECO. *Data_type*: string, optional **Note:** You cannot specify the `-use_gate_array_filler_cells` parameter if you use this parameter.
`-use_gate_array_filler_cells` *GAFillerCells*	
	Specifies a list of Gate Array (GA)-style filler cells that the tool can replace with GA cells during post-mask ECO. You can only use this parameter in a post-mask ECO flow. The new logic instances adding during a post-mask flow must already be present in the netlist. *Data_type*: string, optional **Note:** You cannot specify the `-use_gate_array_cells` parameter if you use this parameter.
`-use_gate_array_filler_groups {true|false}`	
	Specifies that filler cell instances will only be replaced by logic instances that belong to the same GA groups. This option can only be used when there is a filler group created by `create_gate_array_filler_group`. `place_eco -use_gate_array_filler_groups` first finds the optimal location for each instance based on its connectivity, then searches for the nearest GA filler cell that is included in the same filler group, places a GA cell at the GA filler location, and deletes the original GA filler. If the GA cell is smaller than the GA filler, the placer creates a new GA filler instance using the list of GA filler cells in the same filler group and places the filler in the gap. If there is only one filler group created, it equals to `place_eco -use_gate_array_filler_cells`. *Default*: `false` *Data_type*: bool, optional

`-use_spare_cells {true|false}`	Maps unplaced cells to spare cells during placement. First, checks whether all unplaced cells match spare cells. Next, chooses a spare cell and changes the connections from the unplaced cell to the selected spare cell. Finally, deletes the unplaced cell. You can use this parameter only for post-mask ECO. This option observes the power domain when selecting the spare cell and maps an unplaced cell to the spare cell in the power domain, based on the instance name of the cell. For example, if module A/B is defined for power domain PD2, this option will map an unplaced cell A/B/inst1 to a spare cell in module A/B. *Data_type*: bool, optional

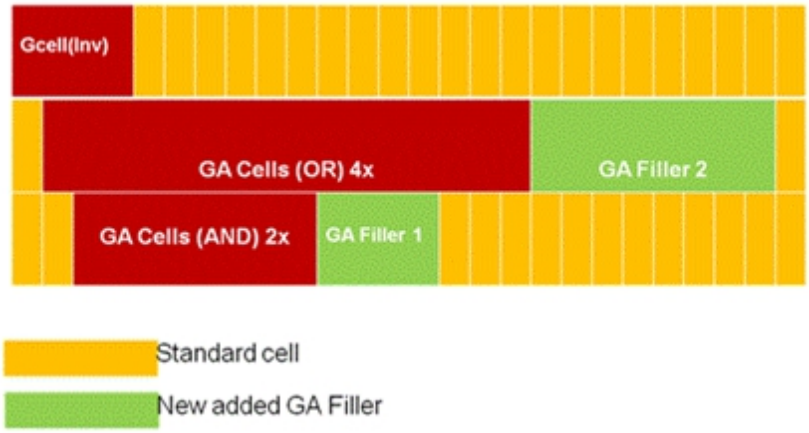
Examples

- The following command performs incremental placement:
  ```
  place_eco -use_spare_cells true
  ```
- The flowing command will place the GA-ECO cells on GACORE site.
  ```
  place_eco -use_gate_array_cells site_name
  ```
- The following command will place GA-ECO cells on specified GA-Fillers and fill remaining gaps with GA-Fillers.
  place_eco -use_gate_array_filler_cells {GAFiller1 GAFiller2 …}

  Before

  After



- The following commands create GA Filler group that associates GA-Function cells with GA-Fillers. Then `place_eco` will place GA-ECO cells on same group GA-Fillers.

```
create_gate_array_filler_group -group GROUP1
-gate_array_cells {GA_Cell1 GA_Cell2… }
-gate_array_fillers {GA_Filler1 GA_Filler2…}
place_eco -use_gate_array_filler_groups true
```

Related Information

- `create_gate_array_filler_group`
- `eco_read_def`
- *ECO Placement* section of the Place Menu chapter of the *Innovus Stylus Common UI Menu Reference*.

read_eco

```
read_eco
[-help]
[-post_mask | -use_gate_array_cells GACoreSiteName]
[-suffix suffix]
[-verbose]
eco_files
```

Reads a file containing ECO directives and applies the changes to the current netlist.

You can use this command at any time after you import the design.

This command also performs consistency checks while the ECO commands are applied and after the changes are complete. It also checks for and observes power domains. This command produces results compatible with `eco_read_def`, so you will use `place_eco` and `route_eco` to complete the ECO process.

The ECO directives do not affect routing, except in the following cases:

- `DELETENET` deletes routing attached to a specified deleted net.
- `DELETEINST` deletes an instance but leaves the wires attached to the instance dangling. You can then repair dangling wires with the router after ECO is complete.

If you want to reroute the entire design, delete routing before running the `read_eco` command.

This command supports ECOs for gate array designs through the `-use_gate_array_cells` and `-post_mask` parameters.

If an error occurs when the `read_eco` command reads the ECO file, the `read_eco` command stops without reading more commands from the file, and writes error messages to the `innovus.log` file.

Note: The first line of the ECO file you read must be the following:

```
FORMATVERSION 2
```

For information on the commands you can include in the ECO command file, see "ECO Directives," in the *Innovus User Guide*.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `read_eco` parameter. For a detailed description of the command and all of its parameters, use the man command: `man read_eco`
`eco_files`	Specifies the name of the file that contains the ECO directives. You can specify more than one file. File names must be separated by one space. *Data_type*: string, required
`-suffix`	Specifies the suffix this command adds to spare cells created by deleting instances. The default suffix is `SPARE`. *Data_type*: bool, optional
`-use_gate_array_cells` *`GACoreSite`*	Uses gate array cells when adding or deleting instances. This parameter applies to gate array designs only. If `ADDINST` is specified in the ECO file, the default behavior for `ADDINST` applies. If `DELETEINST` is included in the ECO file, the software deletes an instance if the instance master uses the same site as the specified *`GACoreSite`*, which can be the name of any gate array site defined in the LEF file. If you do not specify this parameter, `DELETEINST` removes the instance from the netlist. *Data_type*: string, required
`-post_mask`	When adding an instance, the new cell will be unplaced. The unplaced cell will map to the spare cell during placement with the `place_eco -use_spare_cells` command. When the software deletes an instance, it renames the instance to an instance name at the original level of hierarchy, and appends the specified suffix. If you do not specify the `-suffix` parameter, this parameter appends `SPARE`. *Data_type*: bool, optional
`-verbose`	Provides a report for nets without drivers and the instance input pins with no net connections. *Data_type*: bool, optional

Example

- The following command reads `myfile` and runs the ECO commands it contains:

  ```
  read_eco myfile
  ```

- The `read_eco` command issues a summary report on completion, for example:

```
***************** Load ECO Summary *****************
    Number of New Nets:                             0
    Number of New Instances:                        0
    Number of New Ports:                            0
    Number of New Assigns:                          0
    Number of New Feedthroughs:                     0
    Number of Changed Connections:                  0
    Number of Changed Instances:                    0
    Number of Deleted Nets:                         0
    Number of Deleted Instances:                    1
    Number of Deleted Ports:                        0
    Number of Deleted Assigns:                      0
    Number of Deleted Feedthroughs:                 0
```

- The following ECO summary is the output of the `read_eco` command with `-verbose` options:

```
    ***************** Load ECO Summary *****************
    Number of New Nets:                               0
    Number of New Instances:                          1
    Number of New Ports:                              0
    Number of New Assigns:                            0
    Number of New Feedthroughs:                       0
    Number of Changed Connections:                    0
    Number of Changed Instances:                      0
    Number of Deleted Nets:                           0
    Number of Deleted Instances:                      0
    Number of Deleted Ports:                          0
    Number of Deleted Assigns:                        0
    Number of Deleted Feedthroughs:                   0
    Nets With No Driver:
            n_23
            u_counter_1/cnt[30]
```

```
Number of Nets With No Drivers:                    2

Input Instance Pins With No Net Connection:

        A of inst newInst

        TE of inst u_counter_2/ECO_reg_iflg_reg

Number of Input Instance Pins With No Net Connection:   2
```

For an example of an ECO file, see *Example ECO File* in the ECO Directives chapter of the *Innovus Stylus Common UI User Guide.*

Related Information

- eco_read_def
- place_eco
- route_eco
- Example ECO File in the ECO Directives chapter of the *Innovus Stylus Common UI User Guide.*

ungroup

```
ungroup
[-help]
hInst+
[-exclude hinst+]
[-flatten]
[-honor_dont_touch]
[-threshold instance_count]
[-verbose]
[-simple  | -prefix prefix | -escaped ]
```

Ungroups the specified hinsts. Ungrouping dissolves the hinsts and moves the contents into its parent directory. By default, an hinst or an inst inside one of the ungrouped hinsts is renamed by concatenating its name to its parent's name with "_". for example, "ungroup h1", would rename h1/i1 to h1_i1. Name collisions are resolved by appending _1, _2, etc. until a unique name is found. If a given hinst cannot be ungrouped, a warning is issued about why it cannot be ungrouped.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `ungroup` parameter. For a detailed description of the command and all of its parameters, use the man command: `man ungroup`
`hInst+`	Specifies the hinst(s) to be ungrouped. *Data_type*: (hinst)+, required
`-escaped`	Flattened names use escaped hinst names. For example, h1/i1 becomes \h1/i1 in Verilog and h1\/i1 in DEF. *Data_type*: bool, optional
`-exclude` *`hinst+`*	Excludes specified hinst(s) as ungroup targets. *Data_type*: (hinst)+, optional
`-flatten`	Ungroups recursively below the specified hinst(s). *Data_type*: bool, optional
`-honor_dont_touch`	Skips the hinst that is marked as dont touch. *Data_type*: bool, optional
`-prefix` *`prefix`*	Flattened names use a prefix. For example, h1/i1 becomes <prefix>_i1. *Data_type*: string, optional
`-simple`	Flattened names use the base_name. For example, h1/i1 becomes i1. *Data_type*: bool, optional
`-threshold` *`instance_count`*	Ungroups only those hinsts that have an instance count less than or equal to the specified integer. *Data_type*: int, optional
`-verbose`	Prints the names of the hierarchies that are being ungrouped. *Data_type*: bool, optional

Example

- The following command will flatten h1, h2/h3. It will recursively flatten any lower hinsts that have an inst count <= 10 and do not have 'cpu' in their name.

  ```
  >ungroup {h1 h2/h3} -exclude *cpu* -flatten -threshold 10 -prefix test
  ```

- This command ungroups all hinsts matching `h1*` and uses escaped names for the flattened names. This is useful so external SDC files that only refer to inst and pin names will still work but it may cause problems in down-stream tools that do not support escaped names. Note that hpins and hnets of ungrouped hinsts will disappear, so usage of those names will no longer work.

  ```
  >ungroup h1* -escaped
  ```

  An inst named `h1/i1` will become `h1\/i1`. Various commands like `get_db` will still "soft-match" `h1/i1` to the escaped `h1\/i1` name.

Related Information

- `group_instance_suffix attribute` in ECO Without Category Attributes
- `group`

update_inst

```
update_inst
[-help]
-base_cell base_cell
[-dont_snap_to_placement_grid]
[-location {x y}] -name inst_name
[-orient {r0 r90 r180 r270 mx mx90 my my90}]
[-place_status {fixed soft_fixed placed unplaced cover}]
```

Updates the base cell of an existing instance object and returns the updated inst object pointer. The new base cell and existing base cell must have the identical pins.

This command only updates the netlist not the timing graph. For timing eco, `eco_update_cell` is the recommended command to use.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `update_inst` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man update_inst`
`-base_cell` *`base_cell`*	Specifies the base cell to use for the new instance. *Data_type*: (base_cell), required)
`-dont_snap_to_placement_grid`	If `-dont_snap_to_placement_grid` is specified, Innovus places the instance at the specified location even if the location is off the placement grid. This parameter is used with `-location`. *Default*: false *Data_type*: bool, optional
`-name` *`instName`*	Specifies the name of the instance to update. *Data_type*: (inst), required
`-location {`*`x y`*`}`	Specifies the location of the changed instance. *Default:* Default is the same location. *Data_type*: point, optional
`-orient {r0 r90 r180 r270 mx mx90 my my90}`	Specifies orientation of the changed instance. *Default:* Default is the same orientation. *Data_type*: enum, optional
`-place_status {fixed soft_fixed placed unplaced cover}`	Specifies the placement status of the changed instance. *Default*: Default is the same place_status. *Data_type*: enum, optional

Example

- The following command updates the base_cell of `inst i1/i2` from `BUFX2` to `BUFX4`, and then marks it as fixed:
  ```
  update_inst -name i1/i2 -base_cell BUFX4 -place_status fixed
  ```

Related Information

- create_inst
- delete_inst
- eco_update_cell

bitblast_ports

```
bitblast_ports
[-help]
topCell|vCell|hInst+
[-bus busname]
```

Bitblasts the specified port bus (or hport bus) of the specified design, module, or module of the specified hinst, so they are scalars rather than bus-bits. This can be useful to allow optimization for flexibility such as changing the inversion of just some bits of the bus. The name of the bitblasted ports will follow the nomenclature specified by the bit_blasted_port_style global variable. The default style is:

%s_%d

where %s is the bus name, and %d is the bus-bit value. Each bus-bit that is renamed is reported for each module that is changed.

Names used for SDC timing constraints can only be modified if they are loaded into the memory. So, you must ensure that all the timing views are loaded with `set_analysis_view` before running this command. You can either write out the modified SDC files with the new names for later usage or write constraints and Tcl scripts in a way that does not use the original bus-bit names.

There is a limitation that port names used inside a CPF power_intent file cannot be changed. CPF usage assumes that the CPF file is golden and should not be modified. If you need to change names used inside a CPF file, you need to update the CPF file manually, and then read in the new CPF file after running `bitblast_ports`. In contrast, an IEEE1801 power_intent file is allowed to change during the design flow, so changing any port names with bitblast_ports will automatically update the IEEE 1801 names. You can write out the new IEEE1801 file with the updated names.

For non-uniquified netlist, if you bit-blast the hinst, the module and the cloned hinsts also get bit-blasted.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `bitblast_ports`. For a detailed description of the command and all of its parameters, use the man command: `man bitblast_ports`
`topCell|vCell|hInst+`	Specifies the list of design|module|hinst.
`-bus` *`busname`*	Bit-blasts just the specified bus (just “A” not “A[3:0]”), rather than all busses for the module.

Examples

- In the following example, the design top has two-bit input port AI:

```
AI[1:0]
@Innovus> bitblast_ports design:top
#New bitblasted names
Module: top
AI[0]      AI_0
AI[1]      AI_1
@Innovus> get_db ports .name
> AI_0 AI_1
```

- In the following example, hinst1 has two-bit port IN:

```
IN[1:0]
@Innovus> bitblast_ports hinst:top/hinst1 -bus IN
#New bitblasted names
Module: mod1
IN[0]      IN_0
IN[1]      IN_1
@Innovus> get_db hinst:top/hinst1 .hports.name
> hinst1/IN_0 hinst1/IN_1
```

- In the following example, design top has 2 two-bit input ports AI, BI:

```
AI[1:0] BI[1:0]
@Innovus> bitblast_ports design:top -bus AI
#New bitblasted names
Module: mod1
IN[0]      IN_0
IN[1]      IN_1
@Innovus> get_db ports .name
> AI_0 AI_1 BI[0] BI[1]
```

You can bit-blast all ports of two different hinsts:

```
>bitblast_ports {hinst:top/h1 hinst:top/h2} > bitblast.rpt
```

```
>
```

Related Information

- set_analysis_view

delete_unintantiated_modules

```
delete_uninstantiated_modules
[-help]
[-report_only]
```

Deletes uninstantiated modules.

Parameters

Parameters

`-help`	Prints out the command usage. For a detailed description of the command, use the man command: `man delete_uninstantiated_modules`
`-report_only`	Reports all uninstantiated vcells instead of deleting them. All uninstantiated vcells will be deleted if `-reportOnly` is not specified. *Data_type*: bool, optional

Related Information

- ECO Flows chapter in the *Innovus User Guide*

18

Low Power Commands and Attributes

- add_power_switches
- check_power_domains
- check_power_intent
- check_power_switches
- clone_msv_gate
- commit_power_intent
- compare_power_intent
- cut_box_list_from_power_domain
- cut_power_domain_by_overlaps
- delete_power_intent
- delete_power_switches
- get_cpf_user_attributes
- get_pg_pins_use_signal_route
- get_upf_port_attr
- get_valid_power_switch_location
- identify_physical_power_domains
- move_power_switches
- power_intent Category Attributes
- read_power_intent
- read_power_switches
- read_power_switches_for_floorplan
- reorder_power_switch_chains
- report_isolation
- report_partition_boundary_ports
- report_power_domains
- report_power_switches
- report_shifters
- report_voltage
- route_secondary_pg_pins
- set_pg_pins_use_signal_route
- swap_cell_with_always_on_cell
- update_power_domain
- update_power_switches
- write_power_intent

- write_power_switches
- write_power_switches_for_floorplan

add_power_switches

```
  add_power_switches
[-help]
[-1801power_switch_rule_name string]
{-column
-power_domain
-enable_pin_in {listOfPins}
-enable_net_in {listOfNets}
-enable_pin_out {listOfPins}
-enable_net_out {netBaseName}
-global_switch_cell_name cell_syntax
-global_buffer_cell_name cell_syntax
-horizontal_pitch float
[-area {x1 y1 x2 y2}]
[-chain_direction_x LtoR|RtoL]
[-chain_direction_y BtoT|TtoB]
[-switch_module_instance]
[-bottom_offset float]
[-left_offset float]
[-logical]
[-right_offset value]
[-top_offset value]
[-power {(VDD:VDD1 VDD2...)...}]
[-ground {(GND:GND1 GND2...)...}]
[-incremental]
[-width length_in_um]
[-height length_in_um]
[-orient orientation_syntax]
[-connect_bottom_switch_enable_pins {LtoR | RtoL}
[-parallel_enable]
[-global_pattern pattern_syntax]
[-acknowledge_tree_cell cell_name]
[-acknowledge_tree_hinst hier_instance]
[-back_to_back_chain {LtoR | RtoL}]
[-loopback_at_end]
[-max_chain_depth integer]
[-max_distance distance_in_microns]
[-max_fanout integer]
[-snap_to_nearest]
[-skip_rows]
[-top_down]
[-place_by_net_wire_intersection {h_net h_layer v_net v_layer}]
[-place_adjustment_x {delta_in_x}]
[-place_adjustment_x_y {delta_in_x delta_in_y}]
[-place_adjustment_y {delta_in_y}]
[-report_file filename]
[-chain_by_row]
[-checker_board]
[-no_double_height_check]
[-tap_from_nearest_enable_net]
[-allow_partial_stripe_row_overlap]
[-skip_x]
[-skip_y]
[-honor_non_regular_pitch_stripe]
[-no_fixed_std_cell_overlap]
```

```
[-place_under_vertical_net {v_net v_layer}]
[-no_row_verify]
|-ring
-power_domain power_domain_name
-enable_pin_in {listOfPins}
-enable_net_in {listOfNets}
-global_switch_cell_name cell_sytax
-switch_module_instance instanceName
-vertex {x1 y1 x2 y1 x2 y2 x3 y2 x3 y3 x1 y3... xn yn}
[-enable_net_out {listOfNets}]
[-enable_pin_out {listOfPins}]
[-power {(VDD:VDD1 VDD2...)...}]
[-ground {(GND:GND1 GND2...)...}]
[-start_enable_chain_at_corner integer or corner_syntax
[-first_instance_name instanceName]
[-distribute]
[-no_filler]
[-incremental]
[-center_tap_pins pinname_list]
[-report_file filename]
[-define_side_list string]
[-bottom_side string]
[-left_side string]
[-horizontal_side string]
[-right_side string]
[-top_side string]
[-vertical_side string]
[-buffer_cell_side_list {cell_syntax ...}]
[-buffer_cell_name_bottom cell_syntax]}
[-buffer_cell_name_horizontal cell_syntax]
[-buffer_cell_name_left cell_syntax]
[-buffer_cell_name_right cell_syntax]
[-buffer_cell_name_top cell_syntax]
[-buffer_cell_name_vertical cell_syntax]
[-global_buffer_cell_name cell_syntax]
[-filler_cell_side_list {cell_syntax ...}]
[-filler_cell_name_bottom cell_syntax]
[-filler_cell_name_horizontal cell_syntax]
[-filler_cell_name_left cell_syntax]
[-filler_cell_name_right cell_syntax]
[-filler_cell_name_top cell_syntax]
[-filler_cell_name_vertical cell_syntax]
[-global_filler_cell_name cell_syntax]
[-top_switch_cell {cell_syntax ...}]
[-switch_cell_side_list {cell_syntax ...}]
[-switch_cell_name_bottom cell_syntax]
[-switch_cell_name_horizontal cell_syntax]
[-switch_cell_name_left cell_syntax]
[-switch_cell_name_right cell_syntax]
[-switch_cell_name_top cell_syntax]
[-switch_cell_name_vertical cell_syntax]
[-switch_cell_side_list {cell_syntax ...}
[-corner_cell_list {listOfCornerCells}]
[-corner_orient_list {orientation_syntax ...]
[-inside_corner_cell_list {cellName ...]
[-r0_corner corner_syntax]
[-r0_inside_corner {LT | TR | RB | BL}
[-r0_side {L | T | R | B}]
[-r0_side_orientation orientation_syntax]
[-switch_pitch value]
[-switch_pitch_bottom value]
```

```
[-switch_pitch_horizontal value]
[-switch_pitch_left value]
[-switch_pitch_right value]
[-switch_pitch_top value]
[-switch_pitch_vertical value]
[-switch_pitch_side_list {value ...}
[-side_offset_list {value ...}]
[-bottom_offset float]
[-horizontal_offset float]
[-left_offset float]
[-right_offset float]
[-top_offset float]
[-vertical_offset value]
[-side_offset_list {value ...}]
[-start_offset value]
[-start_offset_bottom value}
[-start_offset_horizontal value]
[-start_offset_left value]
[-start_offset_right value]
[-start_offset_top value]
[-start_offset_vertical value]
[-side_start_offset_list [value...]
[-force_offset]
[-global_offset float]
[-end_offset value]
[-end_offset_bottom value]
[-end_offset_horizontal value]
[-end_offset_left value]
[-end_offset_right value]
[-end_offset_top value]
[-end_offset_vertical value]
[-side_end_offset_list {value ...}]
[-side_pattern_list {pattern_syntax ...}]
[-bottom_pattern pattern_syntax]
[-left_pattern pattern_syntax]
[-horizontal_pattern pattern_syntax]
[-right_pattern pattern_syntax]
[-top_pattern pattern_syntax]
[-vertical_pattern pattern_syntax]
[-global_pattern pattern_syntax]
[-continue_pattern]
[-side_num_switch_list {integer ...}]
[-bottom_num_switch integer]
[-horizontal_num_switch value]
[-left_num_switch integer
[-right_num_switch integer]
[-top_num_switch integer]
[-vertical_num_switch value]
[-side_orient_list {orientation_syntax ...}]
[-bottom_orient orientation_syntax]
[-horizontal_orient orientation_syntax]
[-left_orient orientation_syntax]
[-right_orient orientation_syntax]
[-top_orient orientation_syntax]
[-vertical_orient orientation_syntax]}
[-total_number_patterns number_of_patterns]
[-corner_cell_list_by_corner corner_cells_name_per_corner]
[-num_switches]
[-no_min_gap_adjustment]
[-get_switch_instances string]
[-power_domain power_domain_name]
```

```
[{-column | ring}]
[-area {x1 y1 x2 y2}]
[-report_file filename]
[-buffer_delay {value_in_seconds | value_in_seconds value_in_seconds}]
[-buffer_switch_same_direction string] [-cell_em string]
[-cell_em value_in_amps]
[-commit_proto {0|1)]
[-device_threshold_voltage value_in_volts]
[-idSat value_in_amps]
[-iLeak value_in_amps]
[-load_capacitance value_in_farads]
[-max_ir_percentage float]
[-max_leakage_current value_in_amps]
[-max_leakage_percentage float]
[-max_ramp_up_current value_in_amps]
[-max_switch_ir value_in_volts]
[-no_pg_decap_estimate string]
[-no_pg_capacitance_estimate string]
[-number_simultaneous_ramp_up_chain integer]
[-offset_x string] [-orient string]
[-pg_capacitance value_in_farads]
[-pg_capacitance_factor value_in_farads]
[-pg_inductance value_in_henrys]
[-proto_report_file fileName]
[-proto_chain_depth string]
[-proto_chain_depth_given_ramp_up_current string]
[-proto_delay_given_ramp_up_current string]
[-proto_ignore_leakage string]
[-proto_max_chain_depth string]
[-proto_min_chain_depth string]
[-proto_number_switches string]
[-proto_ramp_up_time string]
[-proto_sweep_chain_depth {min_integer max_integer increment_integer}]
[-proto_sweep_switch_number {min_integer max_integer increment_integer}]
[-ramp_up_chain_depth integer]
[-ramp_up_rail_voltage_percentage float]
[-ramp_up_time min_value_in_seconds max_value_in_seconds]
[-power_switch_cell fileName]
[-resistance_on {value_in_ohms | value_in_ohms value_in_ohms}]
[-total_leakage_power value_in_watts]
[-total_power value_in_watts]
[-voltage value_in_volts]
[-no_enable_chain]
[-ignore_shifter_rule]
[-ignore_soft_blockage]
[-cpf_power_switch_rule_name rule_name]
[-cell_enable_pin]
[-instance_prefix]
```

The `add_power_switches` command supports five type of cells: switch cells, filler cells, breaker cells, corner cells (for the ring style only), and buffer cells. You can define a cell pattern when you create power switch rings.

The parameters of this command are categorized as below in the Parameters table depending upon their functionality based on how they are to be used:

- Column Type Insertion Parameters
- Power Switch Prototyping Parameters
- Ring Type Insertion Parameters

Column Type Insertion Parameters

Before you insert power switches in a column style, you must assign power switch cells to `CLASS CORE` and the correct `SITE` in the `.lef` file and connect power/ground nets and pins of the power switch cells. The distance between columns of switches (the horizontal pitch value) must be in µm. You might need to use `create_shifter_rows` to create a new set of rows based on the SITE name if the power switch cell in unique.

The column style is favorable for channel-less designs. In this case, all blocks are abutted and there is no space for rings. In ring style, you must have a space greater than the max ring cell height between the power domain and the adjacent power domain. This space is primarily occupied by ring switch cells and power routing.

If you create power switch columns within a specified power domain the distance between the columns of switches is determined by the `-horizontal_pitch`. The first (left-most) column of switches is located `-left_offset` from the power domain's left boundary. The `-enable_net_in`, which might consist of one or more enable nets, drives the `-enable_pin_in` of the first switch cell of each column.

In column style, nets are assigned to pins sequentially if more nets than number of pins are specified. For example, if the list consists of three nets, and there are five columns of switches, net `1` drives the pin of the first switch in column `1` and `4`, net `2` connects to column `2` and `5`, and net `3` only drive the `-enable_pin_in` of the first switch in column `3`.

Power Switch Prototyping Parameters

These are the parameters, which are used for power switch prototyping.

Ring Type Insertion Parameters

If you insert power switches in ring style, the power switches are added around the selected power domain in the minimum gap available. Unless specified, the power switches are added to the top level hierarchy but the user has a choice to specify to which level in the hierachy should the power switches be added. The command creates a switch ring on the top, bottom, left, and right sides of the specified power domain unless you assign a specific side.

Before you insert power switches in a ring style, you must assign power switch cells to `CLASS RING` in the `.lef` file (SYMMETRY X Y r90, recommended, but not required for switch cells) and connect power/ground nets and pins of the power switch cells. You must have existing enable nets to drive the buffer inside the power switch cells and acknowledge nets exiting the power switch cell. You need these net names when you specify `-enable_net_in` and `-enable_net_out`.

Rectilinear power domains are supported in both ring-style and column-style switch insertion.

Regardless of whether you insert columns or a ring, the software ties the `-enable_pin_out` of the first switch cell to the `-enable_pin_in` of the next switch cell in the same column or on the same side of a ring. All the enable pin of the switch cells in the same column/side are daisy-chained together. For the last switch of the column, the `-enable_pin_out` is tied to `-enable_net_out`.

Note: The parameters which are applicable to both Column Type Insertions and Ring Type Insertions are listed in both sections.

The Syntax Conventions and Examples are listed out before apart from the description the the parameters. For conventions and examples, see:

- Syntax Conventions
- Syntax Examples

For Power Switch Chaining Unchaining Instances, see the description of the `reorder_power_switch_chains` command.

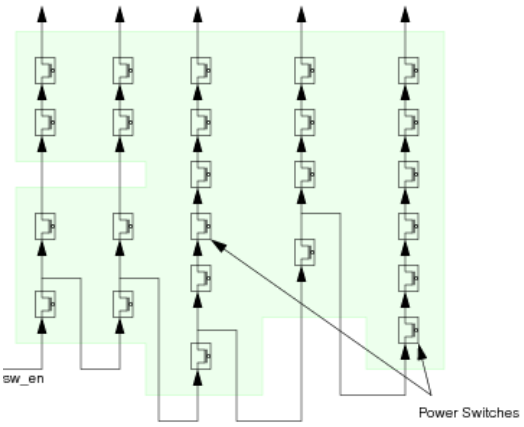
Parameters

-help	Outputs a brief description that includes the type and default information for each `add_power_switches` parameter. For a detailed description of the command and all its parameters, use the `man` command: `man add_power_switches`

`-get_switch_instances` *string*	Returns the existing power switch instances previously added by `add_power_switches`. It returns all the power switches in the current design without any other option, and can be used with `-power_domain`, `-column`, `-ring` and `-area` options to return specific power switches. By default, the command returns a Tcl list. Use the `-report_file` option if you want to list the power switch information into a file. **Example** `add_power_switches -get_switch_instances -power_domain TDSPCore -area {189.0 31.36 318.0 190.36}` *Data_type*: string, optional
Column Type Insertion Parameters	
`-1801power_switch_rule_name` *string*	Specifies the power switch rule name defined in the 1801 power intent file. By default, `add_power_switches` uses the first defined rule. *Data_type*: string, optional
`-acknowledge_tree_cell` *cell_name*	Specifies the name of cell to use in acknowledge tree synthesis. If you specify this parameter but do not specify `-acknowledge_tree_hinst`, the tool builds the acknowledge tree at the top level of the design. This parameter provides control over the time it takes to switch a power domain off and on. *Data_type*: string, optional
`-acknowledge_tree_hinst` *hier_instance*	Inserts an enable net out signal tree of specified cell instances in the specified hierarchical instance in the design. This parameter provides a power domain off and on; and works with option `-acknowledge_tree_cell`. *Data_type*: string, optional
`-allow_partial_stripe_row_overlap`	Allows placement of power switch cells on nearest row location even if crossing stripes do not completely overlap the power switch cells. Use the option with the option `-place_by_net_wire_intersection`. *Data_type*: string, optional
`-area {`*x1 y1 x2 y2*`}`	Defines an area within a power domain where the tool can insert power switches. The option applies to rectangular area. *x1 y1* is the lower left coordinate of the rectangular area. *x2 y2* is the upper right coordinate corner of the rectangular area. You can specify `-left_offset`, `-right_offset`, `-top_offset`, or `-bottom_offset` to offset the switches from the boundaries of the area. **Note:** The power switch instances snap to placement grid so the actual location is slightly different from the specified coordinate. *Data_type*: string, optional

`-back_to_back_chain {LtoR | RtoL}`	Connects the `enable_net_out` at the top of a column to the `enable_net_in` at the top of the next column, and connects the `enable_net_out` at the bottom of the column to the `enable_net_in` at the bottom of the next column. The chain proceeds left to right or right to left. By default, the `enable_net_in` is connected to the bottom of each column, and the `enable_net_out` exits from the top of each column, in parallel. This parameter is optional. *Data_type*: string, optional
`-bottom_offset` *float*	Specifies the distance between the bottom-most switch in the switch column and the bottom fence boundary of the power domain. You can use this parameter along with `-width`, `-height`, and other offsets to define a unique, specific area within the power domain for placing switches. By default, the switch column can abut the bottom fence boundary. For more information, see the Using Pitch Control and Offsets chapter of the *Innovus User Guide* document. *Default:* `0` *Data_type*: string, optional
`-buffer_switch_same_direction` *string*	Used for chaining daisy chain with power switch cell having two enabled pins. The option usage is as follows: • `buffer_switch_same_direction 0` (Default) Input Enable net `EnIn_1` and `EnIn_2` connect first switches at opposite ends; output `EnOut_1` and `EnOut_2` come out as last switch cells in daisy chain. • `buffer_switch_same_direction 1` Input Enable net `EnIn_1` and `EnIn_2` connect first switches at the same end; output `EnOut_1` and `EnOut_2` come out as last switch cells in daisy chain. *Data_type*: string, optional

`-cell_enable_pin`	Specify this option to define/provide the pin names of PSO cells used in `add_power_switches` when the PSO cells pin names are different than given in the parameters `-enable_pin_in` and `-enable_pin_out`. This option can be used with ring as well as column insertion. The option usage is as follows for defining different cell pin signature: `-cell_enable_pin {{cellname {enableInputPinList}\ {enableOutputPinList}}}` In the following example the I/O pin names of switch cells `Cell65_SW_type1` and `Cell65_SW_type2` are different. You can use the `-cell_enable_pin` option to mention the pin name for switch cell `Cell65_SW_type2`. `add_power_switches -column -power_domain PDoff \ -enable_pin_in { SLEEPIN } \ -enable_pin_out { SLEEPOUT } \ -global_switch_cell_name { {Cell65_SW_type1 T1}\ {Cell65_SW_type2 T2} } \ -enable_net_in {Ctrl1_In } \ -enable_net_out { Ctrl1_Out } -global_pattern { T1 T2 } -cell_enable_pin { {Cell65_SW_type2 { SLEEPIN_1 } {SLEEPOUT_1 } }}` In the above example the two pins specified in `-cell_enable_pin` are different from those specified in `-enable_pin_in { SLEEPIN }` and `-enable_pin_out { SLEEPOUT }` *Data_type*: string, optional
`-chain_by_row` *LtoR RtoL*	Gives additional control if you need to enable horizontal signal chaining. For example, in rows. By default, signal chaining is enabled from bottom to top. The signal chaining by this option can be: • LtoR: Signal chaining is enabled from Left to Right direction • RtoL: Signal chaining is enabled from Right to Left direction *Data_type*: string, optional
`-chain_direction_x` *LtoR|RtoL*	Defines the direction of the chain along X axis. *Data_type*: string, optional
`-chain_direction_y` *BtoT|TtoB*	Defines the direction of the chain along Y axis. *Data_type*: string, optional
`-checker_board` *string*	Places the power switches in a checkerboard pattern within a power domain. *Data_type*: string, optional

`-connect_bottom_switch_enable_pins {`*`LtoR	RtoL`*`}`	Connects the first switch of the first column to the enable net specified. The software connects the first switch in the remaining columns to the output enable net of the first switch in the previous column. The following figure illustrates the `LtoR` option (default) for a rectilinear power domain.  **Note:** This parameter applies to power switch column creation only. *Default:* The enable pin of the first switch of each column is connected directly to the enable net signal specified. *Data_type*: string, optional
`-cpf_power_switch_rule_name` *`rule_name`*	Specify this option when there are multiple power switch rules specified for a power domain in CPF. By default, `add_power_switches` uses the first defined rule. *Data_type*: string, optional

`-enable_net_in {listOfnets}`	Specifies the list of input nets that the software attaches to the enable input pins. The order in which you specify these nets determines the sequence in which nets are assigned to pins. Enable nets and pins are connected automatically if they reside at the same level in the hierarchy. If you have already specified `update_power_switch_rule -enable_condition` or `create_power_domain -shutoff_condition` in your CPF file, you do not need to specify this parameter. `update_power_switch_rule -enable_condition` takes precedence over `create_power_domain -shutoff_condition` for enable net In connection. For automatic enable connection to work, you should also specify `update_power_switch_rule -acknowledge_receiver` info in CPF for `enable_net_out` connection. Otherwise you should provide enable net connections using `-enable_net_in` and `-enable_net_out` options. If you specify `-enable_net_in`, it will override `create_power_domain -shutoff_condition`. *Data_type*: string, optional **Note:** You need to specify the `-enable_net_in` and `-enable_pin_in` parameters of the `add_power_switches` command while using the `commit_power_intent -power_switch` parameter for making logical connections between switches.
`-enable_net_out {netBaseName | {listOfNets}}`	Specifies the enable output nets that the software attaches to the enable output pins. The order in which you specify these nets determines the sequence in which nets are assigned to pins. If enable nets and pins reside at the same level in the hierarchy they are connected automatically. The `-enable_pin_out` of the last switch of the last column is tied to `-enable_net_out`. If you have already specified `update_power_switch_rule -acknowledge_receiver` in your CPF file, you do not need to specify this parameter. For automatic enable connection to work, you should also specify `update_power_switch_rule -enable_condition` or `create_power_domain -shutoff_condition` in the CPF file. For `enable_net_in` connection you should provide enable net connections using `-enable_net_in` and `-enable_net_out` options. You must specify *`netBaseName`* for columns or a list of nets for rings. Automatic `enable_net_out` connection works if there is only one enable net out after insertion. *Data_type*: string, optional

`-enable_pin_in {listOfPins}`	Specifies the list of enable input pins of the buffers inside the switch cell. The order in which you specify these nets determines the sequence in which nets are assigned to pins. If enable nets and pins reside at the same level in the hierarchy they are connected automatically.The software ties the `-enable_pin_out` of the first switch cell to the `-enable_pin_in` of the next switch cell in the same ring: All the enable pin of the switch cells in the same column are daisy-chained together. The `-enable_pin_in` of the first switch of the first column, is tied to `-enable_net_in`. *Data_type*: string, optional **Note:** You need to specify the `-enable_net_in` and `-enable_pin_in` parameters of the `add_power_switches` command while using the `commit_power_intent -power_switch` parameter for making logical connections between switches.
`-enable_pin_out {listOfPins}`	Specifies the list of enable output pin(s) of the buffer(s) inside the switch cell. The order in which you specify these pins is important because nets are assigned to pins sequentially. The software ties the `-enable_pin_out` of the first switch cell to the `-enable_pin_in` of the next switch cell in the same column: All the enable pin of the switch cells in the same column are daisy-chained together. For the last switch of the column, the `-enable_pin_out` is tied to `-enable_net_out`. *Data_type*: string, optional
`-first_instance_name` *instanceName*	Specifies the instance name of an existing switch cell in the netlist to use as the first cell in the column or ring. Usually, there is at least one switch instantiated in a module (specified by `-switch_module_instance`) in the netlist. The software creates a unique instance name for each switch inserted. This parameter is optional. If the netlist does not contain switches, then this parameter is invalid. *Data_type*: string, optional
`-global_buffer_cell_name` *cell_syntax*	Specifies the cell name of the buffer cell to insert in the power switch ring surrounding the specified power domain, or in the columns inside the power domain, if there are spaces available. *Data_type*: string, optional
`-global_filler_cell_name` *cell_syntax*	Specifies the cell name of the filler cell to insert in the power switch ring surrounding the specified power domain, or in the columns inside the power domain, if there are spaces available. *Data_type*: string, optional
`-global_pattern` *pattern_syntax*	Specifies a pattern of cells to repeat either in the switch rings or switch columns. The pattern consists of a group of different types of cells in a specified order. *Data_type*: string, optional
`-global_switch_cell_name` *cellName*	Specifies the cell name of the switch cell to insert in the power switch ring surrounding the specified power domain, or in the column inside the power domain. This parameter is required. *Data_type*: string, optional

`-ground {(GND:GND1 GND2...)...}`	Specifies a list of ground nets and pin connections. If `-ground` has multiple ground nets, the software uses the last one in the list for tie-hi/lo connections. For example, specify `-ground (VSS:gnd vss!)` to indicate that the `gnd` and `vss!` pins of all the instances in a given power domain are connected to the `VSS` net. *Data_type*: string, optional
`-height length_in_um`	Specifies the height of an area inside the power domain where switches can be placed. You can use this parameter along with `-width`, `-bottom_offset`, and `-left_offset` to define a unique, specific area in the power domain for placing switches. *Data_type*: string, optional
`-honor_non_regular_pitch_stripe`	Honors the option`-skip_rows` when the pitch/distance between the power stripe specified in `-place_under_vertical_net` option is not regular. This option is related to column insertion. *Data_type*: string, optional
`-horizontal_pitch float`	Specifies the distance between switch columns in µm. This parameter is required. For more information, see the Using Pitch Control and Offsets chapter of the *Innovus User Guide* document. *Data_type*: string, optional
`-ignore_shifter_rule`	Ignores the level shiftier rules, which are applied to the power switches in both CPF and UPF. *Data_type*: string, optional
`-ignore_soft_blockage`	With this option `add_power_switches` can insert power switch cells in areas with soft or partial blockages. By default, `add_power_switches` does not add power switch cells in areas with any type of placement blockage: hard, soft, or partial. *Data_type*: string, optional
`-incremental`	Inserts switches to the power domain without removing the existing switches. *Default:* Running the `add_power_switches` command removes the existing placed switches and inserts new switches. *Data_type*: string, optional
`-instance_prefix prefix`	Adds prefix to power switch instances. **Note**: `-instance_prefix` overwrites the prefix specified in CPF by `update_power_switch_rule`. *Data_type*: string, optional

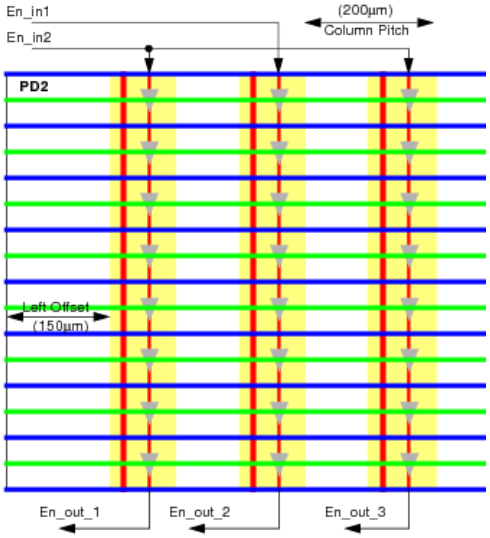
`-left_offset` *float*	Specifies the distance between the left-most switch in the switch column and the right fence boundary of the power domain. You can use this parameter along with `-width`, `-height`, and `-bottom_offset` to define a unique, specific area within the power domain for placing switches. By default, the switch column can abut the right fence boundary. **Note:** When you create a power domain, a `minGap` exists for this power domain, and the switches are inserted within this `minGap`. However, if the specified `minGap` is less than the switch dimension, then the tool automatically extends the `minGap` for this power domain. For more information, see the Using Pitch Control and Offsets chapter of the *Innovus User Guide* document., and refer to commands `-bottom_offset`, `-right_offset`, `-top_offset`, `-vertical_offset`, and `-horizontal_offset`. *Default:* `0` The switches abut the power domain fence if possible. *Data_type*: string, optional
`-logical`	Inserts a logical power switch to the partition block so that the software writes out correct PG direction in dumped PG netlist. *Data_type*: bool, optional
`-loopback_at_end`	Connects the `enable_pin_out` of the last cell in the chain to the `enable_pin_in` of the same cell. This option useful or can be used when a power switch has two `enablePin In` and `Out` pins. *Data_type*: string, optional
`-max_chain_depth` *integer*	Specifies the maximum number of instances in the enable chain. After the specified maximum, `add_power_switches` starts a new enable chain from the next power switch cell. This feature provides fine control over the time it takes to switch a power domain off and on. *Data_type*: string, optional
`-max_distance` *distance_in_microns*	During power switch insertion, `max_distance` is the maximum distance within which the `add_power_switches` looks for the next switch to enable net connections to form an enable signal chain connection. *Data_type*: string, optional
`-max_fanout` *integer*	During power switch insertion, max_fanout is the maximum number of PSO cell enable input pins that a PSO cell enable output pin may connect to. This option is to be used with max_distance. *Data_type*: string, optional
`-no_enable_chain`	Inserts and places power switches without adding enable signal chaining. *Data_type*: string, optional
`-no_fixed_std_cell_overlap`	Avoids placing PSO instances over fixed physical instances already placed on rows. *Data_type*: string, optional

`-no_double_height_check`	Allows double height power switch insertion even if when SITE name is same for double height as well as single height cells. The recommended flow is to use different SITE names for single and double height cells. *Data_type*: string, optional
`-no_row_verify`	Allows you to not verify row coverage at end of a column insertion. By default, the command verifies row coverage. *Data_type*: string, optional
`-offset_x`	Specifies the offset number of intersections in X direction. With this, power switch instances can be inserted in checker board style with specified offset value. **Note:** This must used with `-place_by_net_wire_intersection` and `-skip_x`, options. The value should not be more than the value of `-skip_x`. *Data_type*: string, optional
`-orient` *orientation_syntax*	Specifies the orientation of the switch to be placed in the row. The software checks the specified orientation of the switch against the legal orientation of the row in which the switch is to be placed. The software uses the row site information and the row orientation to determine whether the specified orientation is legal. *Default:* Follows the row orientation. *Data_type*: string, optional
`-parallel_enable`	Connects a power switch's input pin in parallel style rather than daisy-chaining. You do not have to specify `-enable_pin_out` or `-enable_net_out`. *Data_type*: string, optional
`-place_by_net_wire_intersection {`*h_net h_layer v_net v_layer*`}`	Places switches at the intersection of specified horizontal net and vertical net on the respective LEF layers of a pre-existing power grid. You must specify the horizontal net and layer first, then the vertical net and layer. By default, the switches are placed at the bottom-left intersection of the grid. You can offset the switches from the grid intersection by specifying `-place_adjustment_x`, `-place_adjustment_x_y`, or `-place_adjustment_y` in addition to this parameter. *Data_type*: string, optional
`-place_adjustment_x {`*delta_in_x*`}`	Specifies the horizontal offset (in micrometers) from the grid intersection specified by `-place_by_net_wire_intersection`. The value can be positive or negative. *Data_type*: string, optional
`-place_adjustment_x_y {`*delta_in_x delta_in_y*`}`	Specifies the x-y offset (in micrometers) from the grid intersection specified by `-place_by_net_wire_intersection`. The values can be positive or negative. *Data_type*: string, optional
`-place_adjustment_y {`*delta_in_y*`}`	Specifies the vertical offset (in micrometers) from the grid intersection specified by `-place_by_net_wire_intersection`. The values can be positive or negative. *Data_type*: string, optional

`-place_under_vertical_net {vert_net vert_layer}`	Places switches under the vertical net of the specified LEF layer of a pre-existing power grid. By default, the switches are placed at the bottom-left intersection of the grid and row but in addition to this option, you can offset the switches from the row by specifying options like -place_adjustment_x. *Data_type*: string, optional
`-power {(VDD:VDD1 VDD2...)...}`	Specifies a list of power nets and pin connections. For example, specify `-power (Vdd1:Vdd Pwr) (Vdd2:VDD vdd!)` to indicate that the `Vdd` and `Pwr` pins of all the instances in a given power domain are connected to the `Vdd1` net, and the `VDD` and `vdd!` pins are connected to the `Vdd2` net. If `-power` has multiple power nets, the software uses the last one in the list for tie-hi/lo connections. For example, if `-power {(vdd1:VDD1) (vdd2:VDD2)}` is specified, then the software uses net `vdd2` for tie-hi connection. **Note:** You must specify both `-ground` and `-power` to ensure that the instance being moved contains the correct `(net:pin)` connections in the new power domain. *Data_type*: string, optional
`-power_domain power_domain_name`	Specifies the power domain that shuts off through the inserted power switches. *Data_type*: string, optional
`-report_file filename`	Writes switch instances and their corresponding input/output enable signal pin/net connections to the specified file name. *Data_type*: string, optional
`-right_offset`	Specifies the distance between the right-most switch in the switch column and the right fence boundary of the power domain. You can use this parameter along with `-width`, `-height`, and `-left_offset` to define a unique, specific area within the power domain for placing switches. By default, the switch column can abut the right fence boundary. **Note:** When you create a power domain, a `minGap` exists for this power domain, and the switches are inserted within this `minGap`. However, if the specified `minGap` is less than the switch dimension, then the tool automatically extends the `minGap` for this power domain. For more information, see the Using Pitch Control and Offsets chapter of the *Innovus User Guide* document. *Default:* `0` *Data_type*: string, optional
`-skip_x`	Specifies the number of intersections to skip in X direction. **Note:** This option is to be used with `-place_by_net_wire_intersection`. *Data_type*: string, optional

-skip_y	Specifies the number of intersections to skip in Y direction. **Note:** This option is to be used with `-place_by_net_wire_intersection` *Data_type*: string, optional
`-skip_rows` *integer*	Specifies the number of rows to skip when placing a switch in a column. For example, `-skip_rows 2` skips two rows before placing the next switch in a column. You can use this parameter with or without the `-checker_board` parameter. *Data_type*: string, optional
-snap_to_nearest	Snaps the cell placement to the nearest legal location. This parameter can be used when a hard macro blocks part of a column. The switches that would overlap the hard macro blockage are placed at legal locations nearest to the expected original location. You must specify either `-skip_rows` or `-checker_board` in order to create enough space for the cells to snap. If you specify `-skip_rows`, you must set the value greater than zero. *Data_type*: string, optional
`-spacerCellName` *cell_syntax*	Specifies the spacer cells to place on either side of power switch cells within a power domain to prevent the tool from placing standard cells next to the switch cells. *Data_type*: string, optional
`-switch_module_instance` *instanceName*	Specifies module instance where all the switches will be instantiated. This parameter is required. *Data_type*: string, optional

-tap_from_nearest_enable_net	Enables `add_power_switches` to use the enable net from the nearest, already inserted/placed power switch instance. This option can be used during incremental power switch insertions.
	Example :
	`add_power_switches -column \`
	`-power_domain PD1 -enable_pin_in {PinIn} \`
	`-enable_pin_out {PinOut} \`
	`-global_switch_cell_name PSO_SWITCH\`
	`-enable_net_in {pwr_on} -enable_net_out {pwr_ack}\`
	`-horizontal_pitch 100 \`
	`-left_offset 5 \`
	`-area { x1 y1 x2 y2 } -incremental \`
	`-tapfromnearesEnableNet`
	Data_type: string, optional

-top_down	Adds switches from the top down rather than from the bottom up. The following figure shows how the software places switches from top to bottom.  *Data_type*: string, optional
`-top_offset` *value*	Specifies the distance between the top-most switch in the switch column and the top fence boundary of the power domain that shuts off. For more information, see the Using Pitch Control and Offsets chapter of the *Innovus User Guide* document. *Default:* `0` *Data_type*: string, optional
`-width` *length_in_um*	Specifies the width of an area inside the power domain where switches can be placed. Specifies the distance between the bottom-most switch in the switch column and the bottom fence boundary of the power domain that shuts off. You can use this parameter along with `-bottom_offset`, `-height`, and `-left_offset` to define a unique, specific area in the power domain for placing switches. *Data_type*: string, optional
Power Switch Prototyping Parameters	

`-buffer_delay {value_in_seconds | {value_in_seconds value_in_seconds}}`	Specifies the delay of the enable buffer in switch. If the switch is a mother-daughter type (with two enables), you can specify two values corresponding to the stage-1 and stage-2 enable buffers respectively. *Data_type*: string, optional
`-cell_em value_in_amps`	Specifies the maximum current that can pass through the switch due to the electromigration limit of the cell. *Data_type*: string, optional
`-commit_proto string`	Adds the number of switches determined by `-proto_number_switches`. This parameter does not work with any other prototyping commands. *Default*: If you specify `-commit_proto` but do not specify a value, the default is 1. If you do not specify `-commit_proto`, the default is 0. *Data_type*: string, optional
`-device_threshold_voltage value_in_volts`	Specifies the device threshold voltage for ramp up prototyping. If you do not specify this parameter, the tool computes a default value based on the switch characterization parameters. *Data_type*: string, optional
`-idSat {value_in_amps | value_in_amps value_in_amps}`	Specifies the switch current when the switch is operated in saturation mode. If the switch is a mother-daughter type (with two enables), then you can specify two values corresponding to the stage-1 and stage-2 enables respectively. *Data_type*: string, optional
`-iLeak value_in_amps`	Specifies the switch cell leakage current when the switch is off. *Data_type*: string, optional
`-load_capacitance value_in_farads`	Specifies the total captive load of the domain for ramp-up analysis. *Data_type*: string, optional
`-max_ir_percentage float`	Specifies the maximum IR drop allowed inside power domain. *Default*: 2 percent *Data_type*: string, optional
`-max_leakage_current value_in_amps`	Specifies the maximum leakage current constraint of the power domain. *Data_type*: string, optional
`-max_leakage_percentage float`	Specifies the maximum domain leakage constraint as a percentage of the total domain leakage current. *Default*: 50 percent *Data_type*: string, optional
`-max_ramp_up_current value_in_amps`	Specifies the maximum ramp up current value used in chain depth and switch delay prototyping. Use this parameter with `-proto_chain_depth_given_ramp_up_current` or `-proto_delay_given_ramp_up_current`. *Data_type*: string, optional
`-max_switch_ir value_in_volts`	Specifies the maximum voltage drop allowed across the switch. *Data_type*: string, optional

`-netPrefix` *prefix*	Adds a prefix to power switch enable nets.
	For example:
	`add_power_switches -instance_prefix myInst_ -netPrefix myNet_ \`
	`-column -power_domain MPY -enable_net_in swcontrol_2 \`
	`-enable_net_out swack_2 -global_switch_cell_name CDN_COLUMN_SW \`
	`-enable_pin_in A0 -enable_pin_out Z0 -left_offset 10 \`
	`-horizontal_pitch 50 -skip_rows 2 -report_file PSWmpy.rpt \`
	`-switch_module_instance DTMF_INST_4/TDSP_CORE_INST/MPY_32_INST`
	`## From PSWmpy.rpt file`
	`DTMF_INST_4/TDSP_CORE_INST/MPY_32_INST/myInst_psoI_MPY_1_CDN_COLUMN_SW_1415_330`
	`_1 (1415.32 330.87 MX)  # Domain MPY, Column 0, Sequence 1`
	`(A0) :`
	`DTMF_INST_4/TDSP_CORE_INST/MPY_32_INST/myNet_psoPSI_MPY_EnNet__1_1_1415_313_0`
	`(Z0) :`
	`DTMF_INST_4/TDSP_CORE_INST/MPY_32_INST/myNet_psoPSI_MPY_EnNet__1_2_1415_330_0`
	Data_type: string, optional

`-no_pg_capacitance_estimate` *string*	Omits power grid capacitance estimation during ramp up analysis. *Default:* `0` *Data_type*: string, optional
`-no_pg_decap_estimate` *string*	Omits decoupling capacitance estimation during ramp up analysis. *Default:* `0` *Data_type*: string, optional
`-number_simultaneous_ramp_up_chain` *integer*	Specifies the number of simultaneous ramp up chains for ramp up analysis. *Data_type*: string, optional
`-proto_report_file` *fileName*	Specifies the file containing prototyping results. *Data_type*: string, optional
`-pg_capacitance` *value_in_farads*	Specifies the total power grid capacitance for ramp up analysis. *Data_type*: string, optional
`-pg_capacitance_factor` *float*	Specifies the ratio of the power grid capacitance as a factor of the total load capacitance. *Data_type*: string, optional
`-pg_inductance` *value_in_henrys*	Specifies the effective power grid inductance associated with the power grid and package. *Data_type*: string, optional
`-proto_chain_depth` *string*	Performs prototyping on chain depth given maximum ramp up current. You must specify -ramp_up_time when you specify this parameter. *Data_type*: string, optional

`-proto_chain_depth_given_ramp_up_current` *string*	Performs prototyping on chain depth given maximum ramp up current. You must specify `-max_ramp_up_current` when you specify this parameter. *Data_type*: string, optional
`-proto_delay_given_ramp_up_current` *string*	Analyzes optimum ramp up time given maximum ramp up current. You must specify `-max_ramp_up_current` when you use this command. *Data_type*: string, optional
`-proto_ignore_leakage` *string*	Ignores leakage constraints during prototyping. *Data_type*: string, optional
`-proto_max_chain_depth` *string*	Performs maximum chain depth prototyping based on ramp up time. You must specify `-ramp_up_time` when you specify this parameter. *Data_type*: string, optional
`-proto_min_chain_depth` *string*	Performs minimum chain depth prototyping based on ramp up time. You must specify `-ramp_up_time` when you specify this parameter. *Data_type*: string, optional
`-proto_number_switches` *string*	Determines the optimum number of switches based on switch characteristics and domain constraints. *Data_type*: string, optional
`-proto_ramp_up_time` *string*	Performs prototyping on ramp up time based on chain depth. You must specify `-ramp_up_chain_depth` when you specify this parameter. *Data_type*: string, optional
`-proto_sweep_chain_depth {`*min_integer max_integer increment_integer*`}`	Explores various chain depths from minimum to maximum in specified increments, and reports ramp up metrics for each setting. *Data_type*: string, optional
`-proto_sweep_switch_number {`*min_integer max_integer increment_integer*`}`	Explores the number of switches from minimum to maximum in specified increments, and reports corresponding power domain operation metrics. *Data_type*: string, optional
`-ramp_up_chain_depth` *integer*	Specifies the chain depth used in ramp up time prototyping. Specify this parameter when you specify `-proto_ramp_up_time`. *Data_type*: string, optional
`-ramp_up_rail_voltage_percentage` *float*	Specifies the percentage of the full domain voltage upon which the power domain is considered operable when ramping up. *Default*: 90 percent *Data_type*: string, optional
`-ramp_up_time {`*min_value_in_seconds max_value_in_seconds*`}`	Specifies the minimum and maximum ramp up time settings for chain depth prototyping. This parameter is required for `-proto_chain_depth`, `-proto_min_chain_depth`, and `-proto_max_chain_depth`. *Data_type*: string, optional

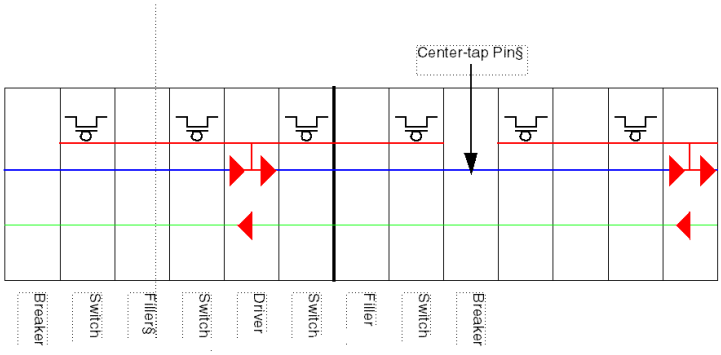
`-power_switch_cell` *`fileName`*	Specifies the file containing the switch characterization data. The file contains a list of characterization data per cell, on one line. If you do not put all of the information for a cell on one line, the tool issues an error message. The format is as follows:
	`CELL cell_name SUPPLY pin_name SWITCHED pin_name RON value_in_ohms idSat value_in_mA iLeak value_in_mA`
	`CELL2...`
	For power switch prototyping, the tool only reads and uses the first line of the file and uses only one cell type.
	Data_type: string, optional

`-resistance_on` *`value_in_ohms`*	Specifies the resistance when the switch is operated in linear mode. If the switch is a mother-daughter type (with two enables), you can specify two values corresponding to the stage 1 and stage 2 enables respectively. *Data_type*: string, optional
`-total_leakage_power` *`value_in_watts`*	Specifies the total leakage power of the power domain before switches are added. *Data_type*: string, optional
`-total_power` *`value_in_watts`*	Specifies the total power consumed by the domain. If you do not specify this parameter, `report_power` computes power consumption. *Data_type*: string, optional
`-voltage` *`value_in_volts`*	Specifies the primary rail voltage of the power domain. *Data_type*: string, optional
Ring Type Insertion Parameters	
`-1801power_switch_rule_name` *`string`*	Specifies the power switch rule name defined in the 1801 power intent file. By default, `add_power_switches` uses the first defined rule. *Data_type*: string, optional

`-bottom_num_switch` *integer*	Specifies the number of switches to insert on the bottom side(s) of the power domain. **Note:** This parameter applies to power switch ring creation only. Three scenarios can occur: • If the spaces available match the number specified, the software fills these spaces with switches. • If the number exceeds the spaces available, the software issues a warning message. • If the number is less than the spaces available, the tool inserts the switches, then inserts filler cells to fill the space. Also, if you specify `f` instead of an integer, the tool automatically fills the bottom side with switches and filler cells. This parameter is optional. See `-horizontal_num_switch`, `-vertical_num_switch`, `-left_num_switch`, `-right_num_switch`, and `-top_num_switch`. *Default:* `f` *Data_type*: string, optional
`-side_num_switch_list` `{`*integer ...*`}`	Specifies the list of switches to insert at the side(s) of the power domain. *Data_type*: string, optional
`-bottom_offset` *float*	Specifies distance from the bottom edge outside of the power domain fence and the bottom side(s) of the power switch ring. See `-horizontal_offset`, `-vertical_offset`, `-left_offset`, `-right_offset`, and `-top_offset`. **Note:** When you create a power domain, a `-min_gap` exists for this power domain, and the switches are inserted within this `-min_gap`. However, if the specified `-min_gap` is less than the switch dimension, then the tool automatically extends the `-min_gap` for this power domain. For more information, see the Using Pitch Control and Offsets chapter of the *Innovus User Guide* document. *Default:* `0` *Data_type*: string, optional
`-bottom_orient` *orientation_syntax*	Specifies the cell orientation for all bottom side(s) of the power switch ring. *Data_type*: string, optional
`-bottom_pattern` *pattern_syntax*	Defines a pattern for the bottom side(s) of the ring only. *Data_type*: string, optional

`-bottom_side` *string*	Inserts switches on the bottom side(s) of the power domain. If you do not specify a side, the software creates a full power switch ring around the specified power domain. If you specify a side, the software inserts switches on that side. In this case, the tool sets `-bottom_side` to 1 and sets the other sides to `0`. You can specify multiple sides and the software inserts switches only on the sides you specify. The tool sets the specified sides to `1` and sets the unspecified sides to `0`. *Default:* `1` *Data_type*: string, optional
`-buffer_cell_name_bottom` *cell_syntax*	Specifies the buffer cells to use on the bottom side(s) of the switch ring. *Data_type*: string, optional
`-buffer_cell_name_horizontal` *cell_syntax*	Specifies the buffer cells to use on the horizontal side(s) of the switch ring. *Data_type*: string, optional
`-buffer_cell_name_left` *cell_syntax*	Specifies the buffer cells to use on the left side(s) of the switch ring. *Data_type*: string, optional
`-buffer_cell_name_right` *cell_syntax*	Specifies the buffer cells to use on the right side(s) of the switch ring. *Data_type*: string, optional
`-buffer_cell_name_top` *cell_syntax*	Specifies the buffer cells to use on the top side(s) of the switch ring. *Data_type*: string, optional
`-buffer_cell_name_vertical` *cell_syntax*	Specifies the buffer cells to use on the vertical side(s) of the switch ring. This parameter applies to power switch ring creation only. *Data_type*: string, optional

`-buffer_cell_side_list {`*cell_syntax …*`}`	Specifies that the buffer cells are inserted on each side of a power domain. The first value of the list indicates the first side related to `Corner 0`, which is the lower left corner of the ring. If the `-counterclockwise` parameter is specified, the first side is considered from from `Corner 0` in the counterclockwise direction. The following example shows the buffer cells specified for an eight-sided power switch ring:
	`-define_side_list {1 0 1 0 1 0 1 0}`
	`-buffer_cell_side_list {cell1 cell1 cell2 cell2 cell3 cell3 cell4 cell4}`
	The `cell1` value is used for the first side. Specify `cell1` again to provide a valid cell type, even when there are no cells on the second side.
	Note: A space must be inserted between the list values. Additionally, only a valid cell name must be specified for each side, even if no cells can be specified for a side. The tools checks whether the number of specified cells is the same as the number specified with `-define_side_list`. In case of any discrepancy, an error message is displayed.
	Data_type: string, optional

`-cell_enable_pin` *string*	Specify this option to define/provide the pin names of PSO cells used in `add_power_switches` when the PSO cells pin names are different than given in the parameters `-enable_pin_in` and `-enable_pin_out`. This option can be used with ring as well as column insertion. The option usage is as follows for defining different cell pin signature: `-cell_enable_pin {{cellname {enableInputPinList}\ {enableOutputPinList}}}` In the following example the I/O pin names of switch cells `Cell65_SW_type1` and `Cell65_SW_type2` are different. You can use the `-cell_enable_pin` option to mention the pin name for switch cell `Cell65_SW_type2`. `add_power_switches -ring -power_domain PDoff \ -enable_pin_in { SLEEPIN } \ -enable_pin_out { SLEEPOUT } \ -global_switch_cell_name { {Cell65_SW_type1 T1}\{ Cell65_SW_type2 T2} } \ -enable_net_in {Ctrl1_In } \ -enable_net_out { Ctrl1_Out } -global_pattern { T1 T2 }- cell_enable_pin { {Cell65_SW_type2 { SLEEPIN_1 } { SLEEPOUT_1 } }}` In the above example the two pins specified in `-cell_enable_pin` are different from those specified in `-enable_pin_in { SLEEPIN }` and `-enable_pin_out { SLEEPOUT }` *Data_type*: string, optional
`-center_tap_pins` *pinname_list*	Enables the center tap capability, and specifies the pins to use as center taps. You must use breaker cells to terminate the buffer signal propagation on both sides of the center tap buffer. The following figure shows that, depending on the specified pin name list, the software assigns a pin name to the enable net. This feature can be useful for debugging the enable signal. **Note:** This parameter applies to power switch ring creation only.  *Data_type*: string, optional
`-continue_pattern`	Continues a specified pattern from one side of the ring to the next side. You must also specify `-global_pattern` when you use this parameter. *Default:* If you do not use this parameter, the software stops the pattern at the end of a side and restarts the pattern from the beginning on the next side. *Data_type*: string, optional

`-corner_cell_list {listOfCornerCells}`	Specifies the list of available corner cells that can be used by the tool to fill the corners of the ring. This parameter is optional. If user specifies only one corner cell, that same cell is used on all four corners with the appropriate orientation for a rectangular power domain. *Data_type*: string, optional

`-corner_cell_list_by_corner corner_cells_name_per_corner`	Specifies the corner cells to be used at different corners. This is applicable for ring only. This option is optional.
	In an example of a rectangular power domain with four corners and two different corner cells, we name the cells: `CORNER_cell1` and `CORNER_cell2` using the option `-corner_cell_list` and then we map them to each corner as shown below:
	`add_power_switches -power_domain PSO -ring \`
	`-global_switch_cell_name {{VDDISWITCH } \`
	`-global_filler_cell_name {{FILLERCELL } \`
	`-specifySides {1 1 1 1 } \`
	`-global_offset 4 \`
	`-corner_cell_list { CORNER_cell1 CORNER_cell2 } \`
	`-corner_cell_list_by_corner { CORNER_cell1\ CORNER_cell2 CORNER_cell1 CORNER_cell2 }\`
	`.... \`
	Data_type: string, optional

`-corner_orient_list {orientation_syntax …}`	Specifies cell orientations for the tool to use for each corner of switch ring, starting with Corner 0 (the lower left corner). List the orientations in a list, with the first orientation corresponding to the first side, the second orientation to the second side, and so on. If you do not specify this parameter, the software determines the corner cell orientations. *Data_type*: string, optional
`-cpf_power_switch_rule_name rule_name`	Specify this option when there are multiple power switch rules specified for a power domain in CPF. By default, `add_power_switches` uses the first defined rule. *Data_type*: string, optional
`-device_threshold_voltage value_in_volts`	Specifies the device threshold voltage for ramp up prototyping. *Data_type*: string, optional
`-distribute`	Distributes switches of different types as evenly as possible on a ring side. For example, if there are 40 available slots, and there are 20 switches and 20 fillers, the software will alternate switches and fillers in the available slots. Use this option together with `-horizontal_num_switch`, `-vertical_num_switch`, `-left_num_switch`, `-right_num_switch`, `-top_num_switch`, or `-bottom_num_switch`. *Data_type*: string, optional

`-enable_net_in {listOfnets}`	Specifies the list of input nets that the software attaches to the enable input pins. The order in which you specify these nets determines the sequence in which nets are assigned to pins. Enable nets and pins are connected automatically if they reside at the same level in the hierarchy. If you have already specified `update_power_switch_rule -enable_condition` or `create_power_domain -shutoff_condition` in your CPF file, you do not need to specify this parameter. `update_power_switch_rule -enable_condition` takes precedence over `create_power_domain -shutoff_condition` for enable net In connection. For automatic enable connection to work, you should also specify `update_power_switch_rule -acknowledge_receiver` info in CPF for `enable_net_out` connection. Otherwise you should provide enable net connections using `-enable_net_in` and `-enable_net_out` options. If you specify `-enable_net_in`, it will override `create_power_domain -shutoff_condition`. *Data_type*: string, optional **Note:** You need to specify the `-enable_net_in` and `-enable_pin_in` parameters of the `add_power_switches` command while using the `commit_power_intent -power_switch` parameter for making logical connections between switches.
`-enable_net_out {`*`netBaseName`*` | {listOfNets}}`	Specifies the enable output nets that the software attaches to the enable output pins. The order in which you specify these nets determines the sequence in which nets are assigned to pins. If enable nets and pins reside at the same level in the hierarchy they are connected automatically. For the last switch of the ring, the `-enable_pin_out` is tied to `-enable_net_out`. If you have already specified `update_power_switch_rule -acknowledge_receiver` in your CPF file, you do not need to specify this parameter. For automatic enable connection to work, you should also specify `update_power_switch_rule -enable_condition` or `create_power_domain -shutoff_condition` in the CPF file. For `enable_net_in` connection you should provide enable net connections using `-enable_net_in` and `-enable_net_out` options. You must specify *`netBaseName`* for columns or a list of nets for rings. Automatic `enable_net_out` connection works if there is only one enable net out after insertion. *Data_type*: string, optional

`-enable_pin_in {listOfPins}`	Specifies the list of enable input pins of the buffers inside the switch cell. The order in which you specify these nets determines the sequence in which nets are assigned to pins. If enable nets and pins reside at the same level in the hierarchy they are connected automatically.The software ties the `-enable_pin_out` of the first switch cell to the `-enable_pin_in` of the next switch cell in the same ring: All the enable pin of the switch cells in the same column are daisy-chained together. For the first switch of the ring, the `-enable_pin_in` is tied to `-enable_net_in`. *Data_type*: string, optional **Note:** You need to specify the `-enable_net_in` and `-enable_pin_in` parameters of the `add_power_switches` command while using the `commit_power_intent -power_switch` parameter for making logical connections between switches.
`-enable_pin_out {listOfPins}`	Specifies the list of enable output pin(s) of the buffer(s) inside the switch cell. The order in which you specify these pins is important because nets are assigned to pins sequentially. The software ties the `-enable_pin_out` of the first switch cell to the `-enable_pin_in` of the next switch cell in the same column: All the enable pin of the switch cells in the same column are daisy-chained together. For the last switch of the column, the `-enable_pin_out` is tied to `-enable_net_out`. *Data_type*: string, optional
`-end_offset value`	Places the last switch on every side of the power domain at the specified distance from the end of each side. The side end is not necessarily determined by the power domain boundary. You can use `-left_offset`, `-right_offset`, `-top_offset`, `-bottom_offset`, `-horizontal_offset`, and/or `-vertical_offset` to determine the side end relative to the power domain boundary. For more information, see the Using Pitch Control and Offsets chapter of the *Innovus User Guide* document. *Default*: `0` *Data_type*: string, optional
`-end_offset_bottom value`	Places the last switch on bottom side of the power domain at the specified distance from end of the side. The side end is not necessarily determined by the power domain boundary. You can use `-left_offset`, `-right_offset`, and/or `-vertical_offset` to determine the side end relative to the power domain boundary. • If the value is positive, the software places the switch to the right of the end location. • If the value is negative, the software places the switch to the left of the end location. For more information, see the Using Pitch Control and Offsets chapter of the *Innovus User Guide* document. *Default:* `0` *Data_type*: string, optional

`-end_offsetHoriztonal` *value*	Places the last switch on the horizontal side of the power domain at the specified distance from end of the side. The side end is not necessarily determined by the power domain boundary. You can use `-left_offset`,`-right_offset`, and/or `-vertical_offset` to determine the side end relative to the power domain boundary. For more information, see the Using Pitch Control and Offsets chapter of the *Innovus User Guide* document. *Default:* `0` *Data_type*: string, optional
`-end_offset_left` *value*	Places the last switch on the left side of the power domain at the specified distance from end of the side. The side end is not necessarily determined by the power domain boundary. You can use `-top_offset -bottom_offset` and/or `-horizontal_offset` to determine the side end location relative to the power domain boundary. • If the value is positive, the software places the switch below the end location. • If the value is negative, the software places the switch above the end location. For more information, see the Using Pitch Control and Offsets chapter of the *Innovus User Guide* document. *Default:* `0` *Data_type*: string, optional
`-end_offset_right` *value*	Places the last switch on the right side of the power domain at the specified distance from end of the side. The side end is not necessarily determined by the power domain boundary. You can use the `-top_offset`, `-bottom_offset`, or `-horizontal_offset` to determine the side end relative to the power domain boundary. • If the value is positive, the software places the switch above the end location. • If the value is negative, the software places the switch below the end location. For more information, see the Using Pitch Control and Offsets chapter of the *Innovus User Guide* document. *Default:* `0` *Data_type*: string, optional

`-end_offset_top` *value*	Places the last switch on the top side of the power switch ring at the specified distance from end of the side. The side end is not necessarily determined by the power domain boundary. You can use `-left_offset`, `-right_offset` or `-vertical_offset` to determine the side end relative to the power domain boundary. • If the value is positive, the software places the switch to the left of the end location. • If the value is negative, the software places the switch to the right of the end location. For more information, see the Using Pitch Control and Offsets chapter of the *Innovus User Guide* document. *Default:* `0` *Data_type*: string, optional
`-end_offset_vertical` *value*	Places the last switch on the vertical sides of the power switch ring at the specified distance from end of the side. The side end is not necessarily determined by the power domain boundary. You can use `-top_offset`, `-bottom_offset`, and/or `-horizontal_offset` to determine to the side end relative to the power domain boundary. For more information, see the Using Pitch Control and Offsets chapter of the *Innovus User Guide* document. *Default:* `0` *Data_type*: string, optional
`-filler_cell_name_bottom` *cell_syntax*	Specifies the filler cell used for the bottom side(s) of the ring. You can specify a different filler cell for each side of the ring. This parameter is optional. *Default:* The filler cell specified by `-global_filler_cell_name`. *Data_type*: string, optional
`-filler_cell_name_horizontal` *cell_syntax*	Specifies the filler cell used for the horizontal side(s) of the ring. You can specify a different filler cell for each side of the ring. This parameter is optional. *Default:* The filler cell specified by `-global_filler_cell_name`. *Data_type*: string, optional
`-filler_cell_name_left` *cell_syntax*	Specifies the filler cell used for the left side(s) of the ring. You can specify a different filler cell for each side of the ring. This parameter is optional. *Default:* The filler cell specified by `-global_filler_cell_name`. *Data_type*: string, optional
`-filler_cell_name_right` *cell_syntax*	Specifies the filler cell used for the right side(s) of the ring. You can specify a different filler cell for each side of the ring. This parameter is optional. *Default:* The filler cell specified by `-global_filler_cell_name`. *Data_type*: string, optional

`-filler_cell_name_top` *cell_syntax*	Specifies the filler cell used for the top side(s) of the ring. You can specify a different filler cell for each side of the ring. This parameter is optional. *Default:* The filler cell specified by `-global_filler_cell_name`. *Data_type*: string, optional
`-filler_cell_name_vertical` *cell_syntax*	Specifies the filler cell used for the vertical side(s) of the ring. You can specify a different filler cell for each side of the ring. This parameter is optional. *Default:* The filler cell specified by `-global_filler_cell_name`. *Data_type*: string, optional

`-filler_cell_side_list {`*cell_syntax …*`}`	Specifies the filler cells to insert on each side of a power domain. The first member of the list indicates the first side relative to Corner 0, which is the lower left corner of the ring.
	The following example shows the filler cells specified for an eight-sided power switch ring:
	`-define_side_list {1 0 1 0 1 0 1 0}`
	`-filler_cell_side_list {cell1 cell1 cell2 cell2 cell3 cell3 cell4\ cell4}`
	`cell1` is used for the first side. You can then specify `cell1` again to provide a valid cell type, even though there can be no cells on the second side.
	Info: You must include a space between each item in the list.
	You *must* specify a valid cell name for each side, even if no cells can be specified for a side. The tools checks that the number of cells you specify is the same number you specified with `-define_side_list`, and issues an error message if it finds a discrepancy.
	Data_type: string, optional

`-first_instance_name` *instanceName*	Specifies the instance name of an existing switch cell in the netlist to use as the first cell in the column or ring. Usually, there is at least one switch instantiated in a module (specified by `-switch_module_instance`) in the netlist. The software creates a unique instance name for each switch inserted. This parameter is optional. If the netlist does not contain switches, then this parameter is invalid. *Data_type*: string, optional
`-force_offset`	Forces the software to place ring switches at the exact offset you specify for all sides. Sides with unspecified offsets are forced to 0.0 µ. When you specify any other offset without specifying `-force_offset`, the tool places switches at the smallest distance that is greater than or equal to the specified offset for each side, in order to maintain a contiguous ring. For more information, see the Using Pitch Control and Offsets chapter of the *Innovus User Guide* document. **Note:** Using this parameter might result in gaps in the ring. *Data_type*: string, optional

`-global_buffer_cell_name` *cell_syntax*	Specifies the cell name of the buffer cell to insert in the power switch ring surrounding the specified power domain, or in the columns inside the power domain, if there are spaces available. *Data_type*: string, optional
`-global_filler_cell_name` *cell_syntax*	Specifies the cell name of the filler cell to insert in the power switch ring surrounding the specified power domain, or in the columns inside the power domain, if there are spaces available. *Data_type*: string, optional
`-global_offset` *value*	Offsets all sides of the power switch ring the specified distance from the power domain boundary. Specify the offset in µm. For more information, see the Using Pitch Control and Offsets chapter of the *Innovus User Guide* document. *Default:* `0` *Data_type*: string, optional
`-global_pattern` *pattern_syntax*	Specifies a pattern of cells to repeat either in the switch rings or switch columns. The pattern consists of a group of different types of cells in a specified order. In the ring style, you can use the `-continue_pattern` parameter to continue the pattern around the corner, from one side of the ring to another. *Data_type*: string, optional
`-global_switch_cell_name` *cellName*	Specifies the cell name of the switch cell to insert in the power switch ring surrounding the specified power domain, or in the column inside the power domain. This parameter is required. *Data_type*: string, optional
`-ground {(`*GND:GND1 GND2*`...)...}`	Specifies a list of ground nets and pin connections. If `-ground` has multiple ground nets, the software uses the last one in the list for tie-hi/lo connections. For example, specify `-ground (VSS:gnd vss!)` to indicate that the `gnd` and `vss!` pins of all the instances in a given power domain are connected to the `VSS` net. *Data_type*: string, optional
`-horizontal_num_switch` *value*	Specifies the number of switches to insert on the horizontal side(s) of the power domain. **Note:** This parameter applies to power switch ring creation only. Three scenarios can occur: • If the spaces available match the number specified, the software fills these spaces with switches. • If the number exceeds the spaces available, the software issues the following error message. • If the number is less than the spaces available, the tool inserts the switches, then inserts filler cells to fill the space. This parameter is optional. See `-left_num_switch`, `-right_num_switch`, and `-top_num_switch`. *Data_type*: string, optional

`-horizontal_pattern` *pattern_syntax*	Specifies a pattern of cells to repeat on the horizontal side(s) of the power domain when switches are inserted. The pattern consists of a group of different types of cells in a specified order. *Data_type*: string, optional
`-horizontal_offset` *value*	Offsets the switch ring from the horizontal edge of the power domain. Specify the offset in µm. For more information, see the Using Pitch Control and Offsets chapter of the *Innovus User Guide* document. *Default:* `0` *Data_type*: string, optional
`-horizontal_orient` *orientation_syntax*	Specifies the cell orientation for all horizontal side(s) of the power switch ring. *Data_type*: string, optional
`-horizontal_side` *string*	Inserts switches on all horizontal side(s) of a power domain. You cannot specify this parameter in conjunction with `-vertical_side`, `-bottom_side`, `-top_side`, `-left_side`, or `-right_side`. *Default:* `1` *Data_type*: string, optional
`-ignore_soft_blockage`	With this option `add_power_switches` can insert power switch cells in areas with soft or partial blockages. By default, `add_power_switches` does not add power switch cells in areas with any type of placement blockage: hard, soft, or partial. *Data_type*: string, optional
`-incremental`	Inserts switches to the power domain without removing the existing switches. *Default:* Running the `add_power_switches` command removes the existing placed switches and inserts new switches. *Data_type*: string, optional
`-inside_corner_cell_list {`*cellName …*`}`	Specifies a list of cells for the tool to choose for the inside corners of the power domain. The following example specifies the corner cells any power switch ring: `-inside_corner_cell_list {cell1 cell2}` **Note:** You do not need to list a specific cell for a specific corner. The tool selects from the list you provide. *Data_type*: string, optional
`-instance_prefix` *prefix*	Adds prefix to power switch instances. The option applies to column and ring insertion.

`-left_offset` *float*	Specifies distance from the right edge outside of the power domain fence and the right side of the power switch ring. See `-horizontal_offset`, `-vertical_offset`, `-right_offset`, `-bottom_offset`, and `-top_offset`. **Note:** When you create a power domain, a `-min_gap` exists for this power domain, and the switches are inserted within this `-min_gap`. However, if the specified `-min_gap` is less than the switch dimension, then the tool automatically extends the `-min_gap` for this power domain. For more information, see the Using Pitch Control and Offsets chapter of the *Innovus User Guide* document, and see `-bottom_offset`, `-right_offset`, `-top_offset`, `-vertical_offset`, and `-horizontal_offset`. *Default:* `0` The switches abut the power domain fence if possible. *Data_type*: string, optional
`-left_num_switch` *integer*	Specifies the number of switches to insert on left side(s) of the power domain. Three scenarios can occur: • If the spaces available match the number specified, then the software fills these spaces with switches. • If the number exceeds the spaces available, the software issues a warning message. • If the number is less than the spaces available, the tool inserts the switches, then inserts filler cells to fill the space. This parameter is optional. See `-bottom_num_switch`,`-right_num_switch`, and `-top_num_switch`. *Default:* `f` *Data_type*: string, optional
`-left_orient` *orienation_syntax*	Specifies the cell orientation for all left side(s) of the power switch ring. *Data_type*: string, optional
`-left_pattern` *pattern_syntax*	Defines a pattern for the left side(s) of the ring only. You can use this parameter with `-bottom_pattern`, `-top_pattern`, and `-right_pattern`. *Data_type*: string, optional
`-left_side` *string*	Inserts switches on the left side(s) of the power domain. If you do not specify a side, the software creates a full power switch ring around the specified power domain. If you specify a side, the software inserts switches on that side. In this case, the tool sets `-left_side` to 1 and sets the other sides to `0`. You can specify multiple sides and the software inserts switches only on the sides you specify. The tool sets the specified sides to `1` and sets the unspecified sides to `0`. *Default:* 1 *Data_type*: string, optional

`-no_filler`	Suppresses the addition of filler cells in power switch rings. *Data_type*: string, optional
`-no_min_gap_adjustment`	Can be used when the user does not want to extend the minimum power domain gaps to form the power switch ring. *Data_type*: string, optional
`-no_enable_chain`	Inserts and places power switches without adding enable signal chaining. *Data_type*: string, optional
`-num_switches`	Defines total number switches to be added around a power domain. This is applicable to ring type insertion only. *Data_type*: string, optional
`-parallel_enable`	Connects a power switch's input pin in parallel style rather than daisy-chaining. You do not have to specify `-enable_pin_out` or `-enable_net_out`. *Data_type*: string, optional
`-power {(`*`VDD:VDD1 VDD2`*`...)...}`	Specifies a list of power nets and pin connections. For example, specify `-power (Vdd1:Vdd Pwr) (Vdd2:VDD vdd!)` to indicate that the `Vdd` and `Pwr` pins of all the instances in a given power domain are connected to the `Vdd1` net, and the `VDD` and `vdd!` pins are connected to the `Vdd2` net. If `-power` has multiple power nets, the software uses the last one in the list for tie-hi/lo connections. For example, if `-power {(vdd1:VDD1) (vdd2:VDD2)}` is specified, then the software uses net `vdd2` for tie-hi connection. **Note:** You must specify both `-ground` and `-power` to ensure that the instance being moved contains the correct `(net:pin)` connections in the new power domain. *Data_type*: string, optional
`-power_domain` *`power_domain_name`*	Specifies the power domain that shuts off through the inserted power switches. *Data_type*: string, optional
`-report_file` *`filename`*	Writes switch instances and their corresponding input/output enable signal pin/net connections to the specified file name. *Data_type*: string, optional

`-right_offset`	Specifies distance from the right edge outside of the power domain fence and the right side of the power switch ring. See `-horizontal_offset, -vertical_offset, -left_offset, -bottom_offset, and -top_offset`. **Note:** When you create a power domain, a `-min_gap` exists for this power domain, and the switches are inserted within this `-min_gap`. However, if the specified `-min_gap` is less than the switch dimension, then the tool automatically extends the `-min_gap` for this power domain. For more information, see the Using Pitch Control and Offsets chapter of the *Innovus User Guide* document. *Default:* `0` *Data_type*: string, optional
`-r0_corner` *corner_syntax*	Specifies the side where the corner cell instance can be placed in the `r0` orientation without rotation. • `UL` = Upper left • `LL` = Lower left • `LR` = Lower right • `UR` = Upper right The orientation rotates the corner cell 90 degrees for each corner, proceeding counterclockwise. *Default:* `UL` The corner cell in the upper left position in the ring is placed with orientation r0. Default orientations: • `UL = r0` • `LL = mx` • `LR = r180` • `UR = my` *Data_type*: string, optional

`-r0_inside_corner {LT | TR | RB | BL}`	Specifies the inside corner where to place the corner cell instance in the r0 orientation. Use this parameter for rectilinear switch rings containing inside corners. • `LT` = Corner of the left and top sides • `TR` = Corner of the top and right sides • `RB` = Corner of the right and bottom sides • `BL` = Corner of the bottom and left sides *Default:* `LT` The corner cell in the left top position in the ring is placed with the orientation r0. • `LT = r0` • `BL = mx` • `RB = r180` • `TR = my` *Data_type*: string, optional
`-r0_side {L | T | R | B}`	Specifies the side where the switch or filler instance can be placed in the `r0` orientation without rotation. The command adjusts the orientation of the remaining sides so that the switches form a continuous ring. The orientation rotates 90 degrees for each side, proceeding counterclockwise. • `L` = Left side • `T` = Top side • `R` = Right side • `B` = Bottom side The orientation rotates 90 degrees for each side, proceeding counterclockwise. *Default:* `T` The switch and filler cells in the top row are placed with orientation `r0`. Default orientations: • `T = r0` • `L = r90` • `B = r180` • `R = r270` *Data_type*: string, optional
`-r0_side_orient` *orientation_syntax*	Enables the software to rotate or mirror the switch instances on the `r0` side. The software automatically adjusts the orientation of the other sides of the domain to form a continuous ring. *Data_type*: string, optional
`-ring`	Specifies ring style power switch insertion. Power switches are placed outside the power domain. *Data_type*: bool, optional

`-right_num_switch` *integer*	Specifies the number of switches to insert on right side(s) of the power domain. Three scenarios can occur: • If the spaces available match the number specified, then the software fills these spaces with switches. • If the number exceeds the spaces available, the software issues a warning message. • If the number is less than the spaces available, the tool inserts the switches, then inserts filler cells to fill the space. Also, if you specify `f` instead of an integer, the tool automatically fills the right side with switches and filler cells. This parameter is optional. See `-bottom_num_switch`, `-left_num_switch`, and `-top_num_switch`. *Default:* `f` *Data_type*: string, optional
`-right_orient` *orientation_syntax*	Specifies the cell orientation for all right side(s) of the power switch ring. *Data_type*: string, optional
`-right_pattern` *pattern_syntax*	Defines a pattern for the right side(s) of the ring only. You must use this parameter with `-bottom_pattern`, `-top_pattern`, and `-left_pattern`. *Data_type*: string, optional
`-right_side` *string*	Inserts switches on the right side(s) of the power domain. If you do not specify a side, the software creates a full power switch ring around the specified power domain. If you specify a side, the software inserts switches on that side. In this case, the tool sets `-right_side` to 1 and sets other sides to `0`. You can specify multiple sides and the software inserts switches only on the sides you specify. The tool sets the specified sides to `1` and sets the unspecified sides to `0`. *Default:* `1` *Data_type*: string, optional
`-side_end_offset_list` `{`*value* `...}`	Places the last switch at the specified distance from the end of the sides. Use this parameter with rectilinear power domains to specify a value for each side. For more information, see the Using Pitch Control and Offsets chapter of the *Innovus User Guide* document. *Default:* `0` *Data_type*: string, optional

`-side_offset_list {value …}`	Lists the offset values representing the distance between the power domain fence boundary and each side of the power switch ring. Each value in the list corresponds to a side specified with `-define_side_list`.
	
	The following example show that all sides of an eight-sided power domain can have switches:
	
	`-define_side_list {1 1 1 1 1 1 1 1}`
	
	The following example shows an offset of 8 µm from side1, 8 µm from side2, 8 µm from side 3, 6 µm from side 4, 3 µm from side 5, 3 µm from side 6, 3 µm from side 7, and 6 µm from side 8.
	
	`-side_offset_list {8 8 8 6 3 3 3 6}`
	
	Note: When you create a power domain, a `-min_gap` exists for this power domain, and the switches are inserted within this `-min_gap`. However, if the specified `-min_gap` is less than the switch dimension, then the tool automatically extends the `-min_gap` for this power domain.
	
	Info: You must include a space between each item in the list.
	
	You *must* specify a valid cell name for each side, even if no cells can be specified for a side.
	
	The tools checks that the number of cells you specify is the same number you specified with `-define_side_list`, and issues an error message if it finds a discrepancy.
	
	For more information, see the Using Pitch Control and Offsets chapter of the *Innovus User Guide* document.
	
	Default: `0`
	
	Data_type: string, optional

`-side_orient_list {orientation_syntax …}`	Lists the orientations for cells in each side of the power switch ring. Each orientation in the list corresponds to a side specified
	
	The following example shows which sides of an eight-sided power domain can have switches:
	
	`-define_side_list {1 1 1 1 1 1 1 1 }`
	
	Ring switches can be placed on any side.
	
	The following example shows that cells in side 1 have orientation r90, r180 in side 3, r270 in side5, and r0 in side 7:
	
	`-side_orient_list {r90 r90 r180 r180 r270 r270 r0 r0}`
	
	Info: You must include a space between each item in the list.
	
	You *must* specify a valid cell name for each side, even if no cells can be specified for a side.
	
	The tools checks that the number of cells you specify is the same number you specified with `-define_side_list`, and issues an error message if it finds a discrepancy.
	
	Data_type: string, optional

`-side_pattern_list {pattern_syntax …}`	Specifies the cell pattern per side of a power switch ring. Each pattern in the list corresponds to a side specified with `-define_side_list`.
	The following example shows which sides of an eight-sided power domain can have switches:
	`-define_side_list {1 0 1 0 1 0 1 0}`
	Ring switches can be placed at sides 1, 3, 5, and 7.
	The following example shows that cells in side 1 have orientation r90, r180 in side 3, r270 in side5, and r0 in side 7:
	`-side_pattern_list {{cell1 cell2 {cell3 * 6}} {cell1 cell2 {cell3 * 6}} {cell2 cell3 {cell4 * 6}} {cell2 cell3 {cell4 * 6}} {cell1 cell2 {cell3 * 6}} {cell1 cell2 {cell3 * 6}} {cell2 cell3 {cell4 * 6}}{cell2 cell3 {cell4 * 6}}`
	Info: You must include a space between each item in the list.
	You must include a valid pattern for each side, regardless of whether the side has any switches.
	The tools checks that the number of cells you specify is the same number you specified with `-define_side_list`, and issues an error message if it finds a discrepancy.
	Data_type: string, optional

`-start_offset value`	Places the first cell on every side of the power domain at the specified distance from the beginning of each side. The beginning of each side is not necessarily determined by the power domain boundary. You can use `-left_offset`, `-right_offset`, `-top_offset`, `-bottom_offset`, `-horizontal_offset`, and/or `-vertical_offset` to determine the beginning of each side relative to the power domain boundary. For more information, see the Using Pitch Control and Offsets chapter of the *Innovus User Guide* document. *Default*: 0 *Data_type*: string, optional
`-side_end_offset_list {value ...}`	Places the last cell at the specified distance from the ending location of the sides. Use this parameter with rectilinear power domains to specify a value for each side. *Data_type*: string, optional
`-side_start_offset_list {value ...}`	Places the first cell at the specified distance from the starting location of the sides. Use this parameter with rectilinear power domains to specify a value for each side. For more information, see the Using Pitch Control and Offsets chapter of the *Innovus User Guide* document. *Default:* `0` *Data_type*: string, optional

`-define_side_list` *string*	Specifies the sides of a power switch ring that can and cannot contain ring switches. In the list, 1 indicates that the side can contain ring switches, and 0 indicates that the side cannot contain ring switches. The list must contain either 0 or 1 for each side of the shape. The first member of the list indicates the first side relative to Corner 0, which is the lower left corner of the ring. The following example shows which sides of an eight-sided power domain can have sw itches: `-define_side_list {1 0 1 0 1 0 1 0}` Ring switches can be inserted at sides 1, 3, 5, and 7. **Note:** You must include a space between each item in the list. *Data_type*: string, optional
`-start_enable_chain_at_corner` *integer* or *corner_syntax*	Specifies the corner where the switch ring insertion begins. This parameter can accept corner syntax or an integer, which accommodates more than four corners in a rectilinear power domain. The corner numbering starts with corner 0 at the lower left corner, and proceeds clockwise around the switch ring. For example, `-start_enable_chain_at_corner 1` starts the enable chain at the first corner found in a clockwise direction from corner 0. *Data_type*: string, optional
`-start_offset_bottom` *value*	Places the first cell on the bottom side of the power domain at the specified distance from the beginning location of the side.The beginning location of the side is not necessarily determined by the power domain boundary. You can use `-left_offset`, `-right_offset` or `-vertical_offset` to determine the beginning location of the side relative to the power domain boundary. • If the value is positive, the software places the cell to the left of the starting location. • If the value is negative, the software places the cell to the right of the starting location. For more information, see the Using Pitch Control and Offsets chapter of the *Innovus User Guide* document. *Default:* `0` *Data_type*: string, optional

`-start_offset_horizontal` *value*	Places the first cell on the horizontal side of the power domain at the specified distance from the beginning location of the sides. The beginning location of the side is not necessarily determined by the power domain boundary. You can use `-left_offset`, `-right_offset` or `-vertical_offset` to determine the beginning location of the side relative to the power domain boundary. • If the value is positive, the software places the cell to the left of the starting location. • If the value is negative, the software places the cell to the right of the starting location. For more information, see the Using Pitch Control and Offsets chapter of the *Innovus User Guide* document. *Default:* `0` *Data_type*: string, optional
`-start_offset_left` *value*	Places the first cell on the left side of the power domain at the specified distance from the beginning location of the side. The beginning location of the side is not necessarily determined by the power domain boundary. You can use `-bottom_offset`, `-top_offset`, or`-horizontal_offset` to determine the beginning location of the side relative to the power domain boundary. • If the value is positive, the software places the cell above the starting location. • If the value is negative, the software places the cell below the starting location. For more information, see the Using Pitch Control and Offsets chapter of the *Innovus User Guide* document. *Default:* `0` *Data_type*: string, optional
`-start_offset_right` *value*	Places the first cell on the right side of the power domain at the specified distance from the beginning location of the side. The beginning location of the side is not necessarily determined by the power domain boundary. You can use `-top_offset`, `-bottom_offset`, or `-horizontal_offset` to determine the beginning location of the side relative to the power domain boundary. • If the value is positive, the software places the first cell below the beginning location of the side. • If the value is negative, the software places the first cell above the beginning location of the side. For more information, see the Using Pitch Control and Offsets chapter of the *Innovus User Guide* document. *Default:* `0` *Data_type*: string, optional

`-start_offset_top` *value*	Places the first cell at the specified distance from the starting location of the top offset. Use this parameter with `-left_offset`, `-right_offset`, or `-vertical_offset` to determine the beginning location of the side. • If the value is positive, the software places the first cell to the right of the beginning location of the side. • If the value is negative, the software places the first cell to the left of the beginning location of the side. For more information, see the Using Pitch Control and Offsets chapter of the *Innovus User Guide* document. *Default:* `0` *Data_type*: string, optional
`-start_offset_vertical` *value*	Places the first cell on the vertical sides of the power switch ring at the specified distance from beginning location of the vertical sides. The beginning location of the side is not necessarily determined by the power domain boundary. You can use `-top_offset`, `-bottom_offset`, or `-horizontal_offset` to determine the beginning location of the side relative to the power domain boundary. For more information, see the Using Pitch Control and Offsets chapter of the *Innovus User Guide* document. *Default:* `0` *Data_type*: string, optional
`-top_switch_cell` *cell_syntax*	Specifies a top switch cell for the power switch ring. *Data_type*: string, optional
`-switch_cell_name_bottom` *cell_syntax*	Specifies a different switch cell for the bottom side(s) of the power switch ring. *Default:* The switch cell specified by `-global_switch_cell_name`. *Data_type*: string, optional
`-switch_cell_name_horizontal` *cell_syntax*	Specifies a different switch cell for the horizontal side(s) of a power switch ring. *Default:* The switch cell specified by `-global_switch_cell_name`. *Data_type*: string, optional
`-switch_cell_name_left` *cell_syntax*	Specifies a different switch cell for the left side(s) of the power switch ring. *Default:* The switch cell specified by `-global_switch_cell_name`. *Data_type*: string, optional
`-switch_cell_name_right` *cell_syntax*	Specifies a different switch cell for the right side(s) of the power switch ring. *Default:* The switch cell specified by `-global_switch_cell_name`. *Data_type*: string, optional

`-switch_cell_name_top` *cell_syntax*	Specifies a different switch cell for the top side(s) of the power switch ring. *Default:* The switch cell specified by `-global_switch_cell_name`. *Data_type*: string, optional
`-switch_cell_name_vertical` *cell_syntax*	Specifies a different switch cell for the vertical side(s) of a power switch ring. *Default:* The switch cell specified by `-global_switch_cell_name`. *Data_type*: string, optional

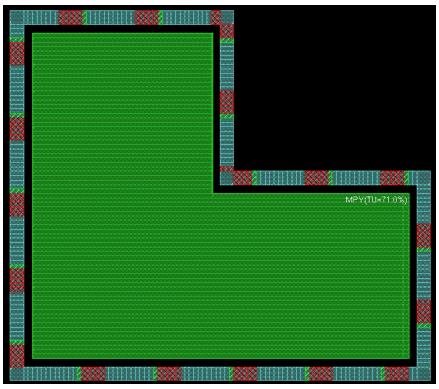
`-switch_cell_side_list {`*cell_syntax …*`}`	Specifies the switch cells to insert on each side of a power domain. The first member of the list indicates the first side relative to Corner 0, which is the lower left corner of the ring.
	The following example shows the switch cells specified for an eight-sided power switch ring:
	`-define_side_list {1 0 1 0 1 0 1 0}`
	`-switch_cell_side_list {cell1 cell1 cell2 cell2 cell3 \ cell3 cell4 cell4}`
	`cell1` is used for the first side. You can then specify `cell1` again to provide a valid cell type, even though there can be no cells on the second side.
	Info: You must include a space between each item in the list.
	You *must* specify a valid cell name for each side, even if no cells can be specified for a side. The tools checks that the number of cells you specify is the same number you specified with `-define_side_list`, and issues an error message if it finds a discrepancy.
	Data_type: string, optional

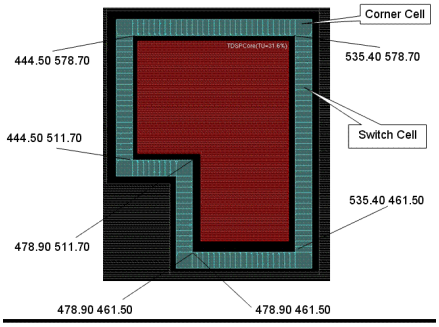
`-switch_module_instance` *instanceName*	Specifies module instance where all the switches will be instantiated. This parameter is required. *Data_type*: string, optional
`-switch_pitch`	Specifies the global switch pitch value. You cannot use this parameter with the `-switch_pitch` parameters (left, right, top, bottom, horizontal, and vertical). For more information, see the Using Pitch Control and Offsets chapter of the *Innovus User Guide* document. *Default:* `0` *Data_type*: string, optional
`-switch_pitch_bottom` *value*	Places the a switch the specified distance from the right edge of the previous cell on the bottom side of the power domain. For more information, see the Using Pitch Control and Offsets chapter of the *Innovus User Guide* document. *Default:* `0` *Data_type*: string, optional

`-switch_pitch_horizontal` *value*	Places the a switch the specified distance from the left edge of the previous cell on the horizontal side of the power domain. For more information, see the Using Pitch Control and Offsets chapter of the *Innovus User Guide* document. *Default:* `0` *Data_type*: string, optional
`-switch_pitch_left` *value*	Places the a switch the specified distance from the bottom edge of the previous cell on the left side of the power domain. For more information, see the Using Pitch Control and Offsets chapter of the *Innovus User Guide* document. *Default:* `0` *Data_type*: string, optional
`-switch_pitch_right` *value*	Places the a switch the specified distance from the top edge of the previous cell on the right side of the power domain. For more information, see the Using Pitch Control and Offsets chapter of the *Innovus User Guide* document. *Default:* `0` *Data_type*: string, optional
`-switch_pitch_top` *value*	Places the a switch the specified distance from the left edge of the previous cell on the top side of the power domain. You can use this parameter with `-switch_pitch*`, `-end_offset*`, and `-*Offset` parameters. For more information, see the Using Pitch Control and Offsets chapter of the *Innovus User Guide* document. *Default:* `0` *Data_type*: string, optional
`-switch_pitch_side_list {`*value* `...}`	Places the a switch the specified distance from the edge of the previous cell on the sides specified in the list. Use this parameter for rectilinear power domains. You can use this parameter with `-switch_pitch*`, `-end_offset*`, and `-*offset` parameters. For more information, see the Using Pitch Control and Offsets chapter of the *Innovus User Guide* document. *Default:* `0` *Data_type*: string, optional
`-switch_pitch_vertical` *value*	Places the a switch the specified distance from the bottom edge of the previous cell on the vertical sides. You can use this parameter with `-switch_pitch*`, `-end_offset*`, and `-*offset` parameters. For more information, see the Using Pitch Control and Offsets chapter of the *Innovus User Guide* document. *Default:* `0` *Data_type*: string, optional

`-top_offset` *float*	Specifies the minimum offset from the top edge outside of the power domain fence boundary where the software can insert switches. See `-bottom_offset`, `-right_offset`, and `-left_offset`. **Note:** When you create a power domain, a `-min_gap` exists for this power domain, and the switches are inserted within this `-min_gap`. However, if the specified `-min_gap` is less than the switch dimension, then the tool automatically extends the `-min_gap` for this power domain. For more information, see the Using Pitch Control and Offsets chapter of the *Innovus User Guide* document. *Default:* `0` The switches abut the power domain fence if possible. Otherwise, the switches are placed as close as possible to the power domain. *Data_type*: string, optional
`-top_num_switch` *integer*	Specifies the number of switches to insert on the top side(s) of the power domain. **Note:** This parameter applies to power switch ring creation only. Three scenarios can occur: • If the spaces available match the number specified, then the software fills these spaces with switches. • If the number exceeds the spaces available, the software issues a warning message. • If the number is less than the spaces available, the tool inserts the switches, then inserts filler cells to fill the space. Also, if you specify `f` instead of an integer, the tool automatically fills the top side with switches and filler cells. This parameter is optional. See `-horizontal_num_switch`, `-vertical_num_switch`, `-bottom_num_switch`,`-left_num_switch`, and `-right_num_switch`. *Default:* `f` *Data_type*: string, optional
`-top_orient` *orientation_syntax*	Specifies the cell orientation for all top side(s) of the power switch ring. *Data_type*: string, optional
`-top_pattern` *pattern_syntax*	Defines a pattern for the top side(s) of the ring only. You must use this parameter with `-horizontal_pattern`, `-vertical_pattern`, `-bottom_pattern`, `-left_pattern`, and `-right_pattern`. *Data_type*: string, optional

`-top_side` *string*	Inserts switches on the top side(s) of the power domain. If you do not specify a side, the software creates a full power switch ring around the specified power domain. If you specify a side, the software inserts switches on that side. In this case, the tool sets `-top_side` to 1 and sets the other sides to `0`. You can specify multiple sides and the software inserts switches only on the sides you specify. The tool sets the specified sides to `1` and sets the unspecified sides to `0`. *Default:* `1` *Data_type*: string, optional
`-total_leakage_power` *value_in_watts*	Specifies the total leakage power of the power domain before switches are added. *Data_type*: string, optional

`-total_number_patterns` *number_of_patterns*	Adds the defined number of patterns around a power domain in a ring-type insertion. The pattern type for the insertion can be defined by the option `-global_pattern`.
	
	 
	
	Example:
	
	`add_power_switches -ring -power_domain MPY -define_side_list {1 1 1\ 1 1 1 } \`
	`-enable_net_in swcontrol_1 -enable_net_in swack_1 \`
	`-global_switch_cell_name {{CDN_RING_SW1 s} {CDN_RING_SW2 b}} \`
	`-global_filler_cell_name {{CDN_RING_FILLER f}} \`
	`-corner_cell_list CDN_RING_CORNER_UL \`
	`-inside_corner_cell_list CDN_RING_CORNER_InCell \`
	`-switch_module_instance DTMF_INST_3 \`
	`-enable_pin_in A0 -enable_pin_out Z0 -side_offset_list { 3 3 3 3 3 \ 3 } \`
	`-global_pattern { {s}*5 {b} } \`
	`-distributePattern 1 \`
	`-continue_pattern 1 \`
	`-total_number_patterns 19 \`
	`-report_file MPY.rpt`
	
	Data_type: string, optional

`-vertex {x1 y1 x2 y1 x2 y2 x3 y2 x3 y3 x1 y3 ... xn yn}`	Allows you to insert power switches in ring style around the power domain by specifying vertex locations around the power domain. In the ring the cells are towards the outer side of the edge.
	 
	
	`add_power_switches \`
	`-ring -vertex \`
	
	`{478.90 461.50 478.90 511.70 444.50 511.70 \ 444.50 578.70 535.40 578.70 535.40 461.50 \ 478.90 461.50} -power_domain TDSPCore...`
	
	Data_type: string, optional

`-vertical_num_switch value`	Specifies the number of switches to insert on all vertical side(s) of the power domain. Three scenarios can occur: • If the spaces available match the number specified, the software fills these spaces with switches. • If the number exceeds the spaces available, the software issues the following error message. • If the number is less than the spaces available, the tool inserts the switches, then inserts filler cells to fill the space. Also, if you specify `f` instead of an integer, the tool automatically fills the bottom side with switches and filler cells. This parameter is optional. See `-left_num_switch`, `-right_num_switch`, and `-top_num_switch`. *Default:* `f` *Data_type*: string, optional
`-vertical_offset value`	Offsets the switch ring from the vertical edge of the power domain. Specify the offset in µm. For more information, see the Using Pitch Control and Offsets chapter of the *Innovus User Guide* document. *Default:* `0` *Data_type*: string, optional
`-vertical_orient orientation_syntax`	Specifies the cell orientation for all vertical side(s) of the power switch ring. *Data_type*: string, optional

`-vertical_pattern` *pattern_syntax*	Specifies a pattern of cells to repeat on all vertical side(s) of the power domain. The pattern consists of a group of different types of cells in a specified order. *Data_type*: string, optional
`-vertical_side` *string*	Inserts ring switches on all vertical side(s) of a power domain. *Default:* 1 *Data_type*: string, optional

Related Topics

- `check_power_switches`
- `delete_power_switches`
- `move_power_switches`
- `read_power_switches`
- `report_power_switches`
- `update_power_switches`
- `write_power_switches`
- Low Power Design chapter in the *Innovus User Guide*

Syntax Conventions

- Cell syntax
  cell_syntax
  cell (single cell)
  {*cell1 cell2 …*} (list of cells)
  {{*cell id*}} (cell with id assigned)
  (cell list with id assigned to each cell)
- Orientation syntax
  orientation_syntax
  r0|r90|r180|r270|mx|my|mx90|my90
- Corner syntax (Ring style only)
  corner_syntax
  LT|TR|RB|BL
- Pattern syntax (Ring style only)
  pattern_syntax
  {id1 id2 {id * n}} (Example: n=integer -> repeat id n times)
  - id, id1, id2 are defined in *cell_syntax*
  - Default id is the same as the cell name if *cell_syntax* does not define it.

Syntax Examples

Guidelines for Examples

switch cell name: CDN_SWT_CELL

Filler cell name: CDN_FLL_CELL

Buffer cell name: CDN_BUF_CELL

Breaker cell name: CDN_BRE_CELL

cell_syntax

cell (single cell): CDN_SWT_CELL

{*cell1 cell2*}: {CDN_SWT_CELL1X CDN_SWT_CELL2X …}

{{*cell id*}}: {{CDN_SWT_CELL S}} <- S can be any alphabetic character from A to Z or a to z, or any combination of alphabetic characters or numerals. The id must always start with an alphabetic character.

Precedence Rules

For the -*direction*_side parameters, the software applies their settings in the following precedence rules:

- -specify_sides
- -top_side, -right_side, -left_side, -bottom_side
- -horizontal_side, -vertical_side

Higher precedence parameters override lower precedent parameters.

Examples

- The following command creates a power switch column within power domain `PD`:

  ```
  add_power_switches -column -power_domain PD -global_switch_cell_name switch_cell -switch_module_instance switch_block \
  -enable_pin_in A0 -enable_pin_out Z0 -enable_net_in {enable_net_on} -enable_net_out {enable_net_out} \
  -left_offset 30 -bottom_offset 0 -horizontal_pitch 240
  ```

  In this example, the software inserts `switch_cell` into power domain `PD`. This switch cell has only one buffer inside. This example specifies one `-enable_net_in (enable_net_on)` and one `-enable_net_out` (`enable_net_out`). The input pin `A0` on the first switch of every column is driven by the net `enable_net_on`. The `enable_pin_out` (`Z0`) drives the `enable_pin_in` pin of the next switch cell in the same column. The nets for this connection are created internally and the names are unique. The `enable_pin_out` pin of the last switch for each column is tied to the net `enable_net_out`. The first switch cell is placed 30 microns from the left side boundary of the selected power domain. The spacing between the columns is 240 microns.

- The following command creates a power switch ring around power domain `PD`.

  ```
  add_power_switches -ring -power_domain PD -switch_module_instance switch_block \
  -enable_pin_in {A0 A1} -enable_pin_out {Z0 Z1} -enable_net_in enIn -enable_net_out enOut \
  -global_switch_cell_name switchCell -global_filler_cell_name filler_cell -first_instance_name u1_switchCell \
  -left_num_switch -right_num_switch 28 -top_num_switch 40 -bottom_num_switch 50
  ```

  In in this example, all `switchCell` switches are instantiated in the `switch_block` module.

  Note: The bottom offset is zero in this example. The first switch cell is placed in the column in or on the bottom-most switch cell row. If the row sites are different for switch cells, you can create them with the `create_shifter_rows -site switch_cell_site` command.

  This example inserts the switches as follows:

- Switches are inserted in a full ring pattern (left, top, right, and bottom)
  The switch ring encircles the `PD` power domain fence. `PD` is the power domain that shuts off.
- A specified number of switches are inserted on each side of the ring. The `-num_switch_left` parameter has a value of `f` (autofill), so the tool fills the left side with as many switches as possible in the available space, then inserts filler cells.
- If the spaces available match the number of switches specified, then the software fills these spaces with switches. If the number of switches is less than the spaces available, the tool inserts the switches, then inserts filler cells `filler_cell` to fill the space. If the number exceeds the spaces available, then the software issues the following error message:

  ```
  ERROR: number of switches exceeded space availability.
  ```

  Also in this example, the original netlist already has a switch instantiated inside the module `switch_block`. The switch is fixed as the first instance in the ring by `-first_instance_name u1_switch_cell`. The tool then generates a unique instance name for each switch.

  You must specify the input and output enable pin names of the switch. You must specify the pins in order because the software attaches them to the enable nets you specify. In this example, there is a loopback type of enable nets, so you must specify one net for input (`-enable_net_in`) and one net for output (`-enable_net_out`). The input net (`enIn`) attaches to the first variable specified in the `-enable_pin_in` list (`A0`) of the first switch, and the output net (`enOut`) attaches to the last variable specified in the `-enable_pin_out` list (`Z1`) of the next switch inserted.

Related Topics

The following sections in the Low Power Design chapter of the *Innovus User Guide*:

- Overview
- Power Domain Shutdown and Scaling
- Power Shutdown Techniques

- Power Switch Optimization

check_power_domains

```
  check_power_domains
[-help]
[-all_inst_in_power_domain]
[-always_on_buffer_type]
[-cell_bind {cell ...}]
[-drc_file string]
[-global_connection [module_name_list] [-always_on_bias]]
[-ignore_net_to_port_iso]
[-ignore_net_to_port_shifter]
[-lib_binding_voltage]
[-lib_cell_binding]
[-nets_missing_iso [fileName]
[-nets_missing_shifter [fileName]
[-place [-place_report report_name]]
[-power_switch]
[-power_within_min_gap]
[-retention {fileName ...}]
```

Verifies whether the power domains are set up correctly in terms of timing library binding and PG connection. This command also checks whether any power domains overlap. The `check_power_domains` command checks that libraries specified for each power domain of each `dc_corner` are complete for each power domain member list. The command also checks if the inserted isolation is consistent with the isolation rule.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `check_power_domains` parameter. For a detailed description of the command and all its parameters, use the `man` command: `man check_power_domains`
`-all_inst_in_power_domain`	Verifies all instances are in the power domain. *Data_type*: bool, optional
`-always_on_bias`	Checks the `always_on_bias` net connection. *Data_type*: bool, optional
`-always_on_buffer_type`	Checks if the AO buffer in each domain has correct type in terms of AO Nwell design and IEEE1801 Nwell specification. *Data_type*: bool, optional
`-cell_bind {cell ...}`	Checks if the specified cells are bound in the domain's library set for all active views and domains. *Data_type*: string, optional

`-drc_file` *fileName*	Creates the Design Rule Checking (DRC) violation markers and save those in the specified file. *Data_type*: string, optional
`-ignore_net_to_port_iso`	Ignores the nets connecting to I/O when verifying isolation using `-nets_missing_iso`. *Data_type*: bool, optional
`-ignore_net_to_port_shifter`	Ignores the nets connecting to I/O when verifying shifter using `-nets_missing_shifter`. *Data_type*: bool, optional
`-lib_binding_voltage`	Checks voltage against the binding library. *Data_type*: bool, optional
`-lib_cell_binding`	Verifies that all cells are bound to correct timing library cells. *Data_type*: bool, optional
`-global_connection` *module_name_list*	
	Verifies that power and ground pins of instances within the power domain are connected to the power and ground nets of that power domain. In addition, this parameter verifies that the tie-high and tie-low connections of an instance are connected to the same power and ground nets of the power domain to which the cell belongs. If you do not specify a list of module names, all modules are verified. *Data_type*: string, optional
`-nets_missing_iso`	Verifies if the nets are missing isolation instances according to the power intent file. It also checks if the isolation uses right type of cell specified in the IEEE1801 `map_isolation_cell` command (if any). *Data_type*: string, optional
`-nets_missing_shifter`	Verifies if the nets are missing level shifter according to the power intent file. It also checks if the level shifter uses right type of cell specified in the IEEE1801 map_level_shifter_cell command (if any). *Data_type*: string, optional
`-place`	Verifies if the instances are placed in the boundary of their power domain. *Data_type*: bool, optional
`-place_report` *report_name*	
	Specifies a name and location for the report generated when you use this command with the `-place` parameter. *Data_type*: string, optional

`-power_switch`	Verifies whether each row in the power domains contain at least one power switch. The command issues a warning message detailing which rows do not contain at least one switch. It also checks if the power switch uses right type of cell specified in the IEEE1801 `map_power_switch_cell` command (if any). *Data_type*: string, optional
`-power_within_min_gap`	
	Verifies that the power domain ring is within the distance specified by the `modifyPowerDomainAttr -minGaps` or `createPowerDomain -minGaps` command. *Data_type*: bool, optional
`-retention {`*`fileName ...`*`}`	
	Specifies a name for storing the verify retention information. It checks if the retention uses the right type of cell specified in the IEEE1801 map_retention_cell command (if any). *Data_type*: string, optional

Example

The following command issues warning messages if instances that are not members of a power domain are placed within that power domain boundary, or if any power domains overlap:

```
check_power_domains -place
```

Related Topics

The following sections in the Low Power Design chapter of User Guide:

- Overview
- Power Domain Shutdown and Scaling
- Multiple supply Voltage Flat Flow
- Verify Power Domains

check_power_intent

```
check_power_intent
[-help]
-1801
-hierarchical
[-out_file string]
[-verbose]
```

Checks the IEEE1801 power intent coding style for the IEEE1801 partitioning. This command also reports potential IEEE1801 coding style, which is unsuitable for partitioning.

Parameters

`-help`	Prints a brief description that includes the type and default information for each `check_power_intent` parameter. For a detailed description of the command and all its parameters, use the man command: `man check_power_intent`
`-1801`	Specifies that the power intent is IEEE 1801. *Data_type*: bool, required
`-hierarchical`	Specifies the input power intent style. *Data_type*: bool, required
`-out_file string`	Specifies the output filename. *Data_type*: string, optional
`-verbose`	Prints the detailed information related to the `check_power_intent` command. *Data_type*: bool, optional

Related Information

- `commit_power_intent`
- `delete_power_intent`
- `read_power_intent`
- `write_power_intent`
- Low Power Design chapter in the *Innovus User Guide*

check_power_switches

```
check_power_switches
[-help]
[-power_domain powerDomain]
[-check_floating_input]
[-check_floating_output]
[-check_floating]
[-check_overlap]
[-check_place]
[-check_loop]
[-check_row_coverage]
[-check_all]
[-highlight]
[-report_longest_chain]
[-starting_net]
```

Verifies power switch placement and enable connections, including the row coverage for column insertion. The command issues a warning message detailing which rows do not contain power switches. The options also enable reporting for the longest signal chain starting from a given enable net. If the start of the enable net is not available, all enable nets of all domains will be used as starting points and all chains will be reported. Both the length of the chain and the switch instances involved in the longest chain will be reported.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `check_power_switches` parameter. For a detailed description of the command and all its parameters, use the `man` command: `man check_power_switches`
`-check_all`	To verify all the checks. *Data_type*: bool, optional
`-check_floating`	To check both floating input and output enable pins. *Data_type*: bool, optional
`-check_floating_input`	To check floating input enable pins. *Data_type*: bool, optional
`-check_floating_output`	To check floating output enable pins. *Data_type*: bool, optional
`-check_loop`	To check for accidentally created enable signal loops. *Data_type*: bool, optional
`-check_overlap`	To check any overlap among the pins. *Data_type*: bool, optional
`-check_place`	To check for unplaced power switches. *Data_type*: bool, optional
`-check_row_coverage`	To check if every row is covered by power switch. *Data_type*: bool, optional
`-highlight`	To select the violation. *Data_type*: bool, optional
`-power_domain powerDomain`	Verify only the specified power domain. *Data_type*: string, optional
`-report_longest_chain`	Reports the longest chain starting from the given enable net. If`-starting_net` is not given, all enable nets of all domains will be used as the starting point and all chains will be reported. The report will include both: the length of the chain, and the switch instances involved in the longest chain. *Data_type*: bool, optional
`-starting_net`	*Data_type*: string, optional

Examples

- The following command verifies the placement status of power switch cells and checks the row coverage by power switch cells for the `powerDomain TDSPCore`:

```
check_power_switches -power_domain TDSPCore -check_place -check_row_coverage
Verifying domain: TDSPCore.
Verifying power switch row coverage ...
Verifying power switch placement status ...
Switch verification completed on domain: TDSPCore.
Number of warnings on domain TDSPCore : 0.
Power switch verification completed with 0 warnings..
1
```

- The following command reports the longest switch chains starting from `net psw_en`:

```
check_power_switches -power_domain PD2 -report_longest_chain -starting_net psw_en
```

Related Topics

- `add_power_switches`
- `delete_power_switches`
- `move_power_switches`
- `read_power_switches`
- `report_power_switches`
- `update_power_switches`
- `write_power_switches`
- Low Power Design chapter in the *Innovus User Guide*

clone_msv_gate

```
clone_msv_gate
[-help]
[-iso]
[-shifter]
[-instances inst1 inst2 .... ]
[-max_fanout integer]
[-max_transition num]
[-max_cap num]
[-honor_pre_placed]
[-honor_dont_touch]
```

Ensures that the cloned isolation or level shifter can drive a group of the receivers and be placed close to this group of receivers. By doing this, the timing QoR can be improved.

Note: The command will not automatically update your CPF file to match the new netlist. It is advised that you run Conformal Low Power to check for any issues. When the enable signal for isolation cells is cloned (creating new ports on the boundary of the power domain) the users may need to update the `create_isolation_rule` in CPF by defining the isolation control differently.

Parameters

`-help`	Prints a brief description that includes the type and default information for each `clone_msv_gate` parameter. For a detailed description of the command and all its parameters, use the man command: `man clone_msv_gate`
`-honor_dont_touch`	Does not clone don't touch instances or cells. The default is to clone don't touch instances. *Data_type*: bool, optional
`-honor_pre_placed`	Does not clone pre-placed (fixed) instances. The default is to clone fixed instances. *Data_type*: bool, optional
`-instances` *inst1 inst2 ...*	Clones specified instances. *Data_type*: string, optional
`-iso`	Clones only the isolation cells. *Data_type*: bool, optional
`-max_cap` *num*	An optional check to clone instances with maximum capacitance violation. The default is not to use `-max_cap` violation. *Data_type*: string, optional
`-max_fanout` *integer*	Specifies the maximum fanout count. *Data_type*: int, optional
`-max_transition` *num*	Cloning will be only on instances with maximum transition time violation. The default is `2ns`. *Data_type*: string, optional
`-shifter`	Clones only the level shifters. *Data_type*: bool, optional

Related Topics

Low Power Design chapter in the *Innovus User Guide*

commit_power_intent

```
  commit_power_intent
[-help]
[-clp_log string]
[-eco string]
[-isolation]
[-keep_rows]
[-level_shifter]
[-power_domain]
[-power_switch]
[-state_retention]
[-verbose]
```

Commits IEEE1801 power intent specifications for the design for use in verification and implementation of the structure and behavior of the design in context of a given power management architecture. The method supports incremental refinement of power intent specifications required for IP-based design flows.

Note: It is needed to commit the power intent using the `commit_power_intent` command after running the `init_design` command for design initialization.

Parameters

`-help`	Prints a brief description that includes the type and default information for each `commit_power_intent` parameter. For a detailed description of the command and all its parameters, use the `man` command: `man commit_power_intent`
`-clp_log` *string*	Specifies the CLP log file for CPF only. *Data_type*: string, optional
`-eco` *string*	Specifies the eco CPF file. **Note:** The `-eco` option is available only for the CPF flow. *Data_type*: string, optional
`-isolation`	Specifies the commit isolation. *Data_type*: bool, optional
`-keep_rows`	Specifies the pre-existing rows to be retained. By default, rows are recreated based on the power domain information. The new rows might not match the original rows. *Data_type*: bool, optional
`-level_shifter`	Specifies the commit level shifter. *Data_type*: bool, optional
`-power_domain`	When specified, the command creates the power domains, makes the pg connection and saves the rules such as level shifter, isolation and state retention rule. **Note**: The command does not commit the rules. *Data_type*: bool, optional
`-power_switch`	Specifies the commit power switch. **Note:** You need to specify the `-enable_net_in` and `-enable_pin_in` parameters of the `add_power_switches` command while using the `commit_power_intent -power_switch` parameter for making logical connections between switches. *Data_type*: bool, optional
`-state_retention`	Specifies the commit state retention. *Data_type*: bool, optional
`-verbose`	Prints the detailed information related to the `commit_power_intent` command. *Data_type*: bool, optional

Related Topics

- `delete_power_intent`
- `init_design`
- `read_power_intent`
- `write_power_intent`
- Low Power Design chapter in the *Innovus User Guide*

compare_power_intent

```
compare_power_intent
[-help]
[-domain_attributes]
[-domain_boundary_port_constraints]
[-domain_map_file file_name]
[-file_prefix prefix]
-golden_1801 file_name
-hier_1801 file_name
[-pin_effective_domain]
[-power_domains list_of_domains]
[-verbose]
```

Compares the golden IEEE1801 and the hierarchical IEEE1801 files generated by the Innovus hierarchical partition flow in terms of the low power database, and reports the mismatched category. If there is no mismatch, the two IEEE1801 files are considered to be equivalent in the Innovus low power flow.

Parameters

`-help`	Prints a brief description that includes the type and default information for each `compare_power_intent` parameter. For a detailed description of the command and all its parameters, use the man command: `man compare_power_intent`
`-domain_attributes`	Compares low power attributes and properties of all the attributes, which include instance count, low power cell count, primary supply, available supply, boundary, port low power constraint, and effective domain counter for the leaf instance pin, and port. *Data_type*: bool, optional
`-domain_boundary_port_constraints`	Compares the domain boundary port IEEE1801 constraint against the golden IEEE1801. *Data_type*: bool, optional
`-domain_map_file` *file_name*	Specifies the domain map file saved during partition. The domain map file maps the golden IEEE1801 domain/supply set/low power signal with the hierarchical IEEE1801 domain/supply set/low power signal. *Data_type*: string, optional
`-file_prefix` *prefix*	Specifies the file prefix for the report files. *Data_type*: string, optional
`-golden_1801` *file_name*	Specifies the golden IEEE1801. *Data_type*: string, optional
`-hier_1801` *file_name*	Specifies the hierarchical IEEE1801 to be compared. *Data_type*: string, optional
`-pin_effective_domain`	Compares only the domain for leaf instance pin and port. *Data_type*: bool, optional
`-power_domains` *list_of_domains*	Specifies the list of specific power domains to be compared. *Data_type*: string, optional
`-verbose`	Displays the detailed information related to each category. *Data_type*: bool, optional

Related Information

- `delete_power_intent`
- `read_power_intent`
- `write_power_intent`
- Low Power Design chapter in the *Innovus User Guide*

cut_box_list_from_power_domain

```
cut_box_list_from_power_domain
[-help]
-power_domain
-box_list
```

Cuts out default power domains from a non-default power domain. This command is mainly used to solve buffering related issues.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `cut_box_list_from_power_domain` parameter. For a detailed description of the command and all its parameters, use the `man` command: `man cut_box_list_from_power_domain`
`-power_domain`	Specifies the power domain in which you want to create the shifter rows. You can create rows for the default power domain in addition to creating rows for the non-default power domains. *Data_type*: string, required
`-box_list`	Specifies the list of boxes to be cut. *Data_type*: string, required

Examples

```
cut_box_list_from_power_domain -power_domain pd -box_list {(list1{ {list2}......}
```

The list is numbers in the format: `{llx1 lly1 urx1 ury1}`

```
cut_box_list_from_power_domain -power_domain 1.0V_Domain \
-box_list {{10 10 20 20} {35 60 45 70}}
```

This will cut box `{10 10 20 20}` and `{35 60 45 70}` from the domain `1.0V_Domain`, and these two boxes will become the default domain.

Related Topics

- `cut_power_domain_by_overlaps`
- Low Power Design chapter in the *Innovus User Guide*

cut_power_domain_by_overlaps

```
cut_power_domain_by_overlaps
power_domain_name
[-help]
```

When you want to floorplan one power domain (inner domain) completely inside the other power domain (outer domain), use this command to cut the outer domain as a hollow or donut shape.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `cut_power_domain_by_overlaps` parameter. For a detailed description of the command and all its parameters, use the `man` command: `man cut_power_domain_by_overlaps`
power_domain_name	Specifies the outer power domain where the nested power domain will reside. *Data_type*: string, required

Related Topics

- `cut_box_list_from_power_domain`
- Low Power Design chapter in the *Innovus User Guide*

delete_power_intent

```
 delete_power_intent
[-help]
[-keep_fence]
```

The command deletes all the power intent data stored by the `commit_power_intent` command in current session and cleans up the design.

Parameters

`-help`	Prints a brief description that includes the type and default information for each `delete_power_intent` parameter. For a detailed description of the command and all its parameters, use the man command: `man delete_power_intent`
`-keep_fence`	Retains domain groups and their fences at the time of removing the the power intent. Use `read_power_intent` and `commit_power_intent -keep_rows` to reload power intent after using this parameter.

Related Topics

- `commit_power_intent`
- `read_power_intent`
- `write_power_intent`
- Low Power Design chapter in the *Innovus User Guide*

delete_power_switches

```
delete_power_switches
[-help]
[-1801_power_switch_rule_name rule_name]
{{-column | -ring}
-power_domain powerDomainName}
[-instance_list {instName ...}]
```

Removes all switches and filler cells that were inserted by the `add_power_switches` command, from either columns or rings, or removes specified power switch defined in CPF.

This command retains and unplaces switches added by `add_power_switches -switch_instances`, whose cells were already instantiated in the `netlist`.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `delete_power_switches` parameter. For a detailed description of the command and all its parameters, use the `man` command: `man delete_power_switches`
`-1801_power_switch_rule_name` *`rule_name`*	Removes power switches based on the specified IEEE 1801 rule. *Data_type*: string, optional
`-column`	Removes all switches and filler cells that were inserted by the `add_power_switches` command, from columns in the specified power domain. *Data_type*: bool, optional
`-instance_list {`*`instName`* `…}`	Specifies the power switch cell instances to delete. *Data_type*: string, optional
`-power_domain` *`powerDomainName`*	Specifies the power domain next to which the columns are placed. *Data_type*: string, optional
`-ring`	Removes all switches and filler cells that were inserted by the `add_power_switches` command, from the ring around the specified power domain. *Data_type*: bool, optional

Examples

- The following command removes instances `psoI_TDSP3_instA` and `psoI_TDSP3_InstB` that were added by the `add_power_switches` command:
  ```
  delete_power_switches -instance_list {psoI_TDSP3_instA psoI_TDSP3_InstB}
  ```
- The following command removes all power switch instances that were inserted by `add_power_switches` in columns inside the power domain `PD2`:
  ```
  delete_power_switches -column -power_domain PD2
  ```

- The following command removes all power switch instances that were inserted by `add_power_switches` in the ring around `PD1`:
  ```
  delete_power_switches -ring -power_domain PD1
  ```

Related Topics

- `add_power_switches`
- `check_power_switches`
- `move_power_switches`
- `read_power_switches`
- `report_power_switches`
- `update_power_switches`
- `write_power_switches`
- Low Power Design chapter in the *Innovus User Guide*

get_cpf_user_attributes

```
get_cpf_user_attributes
[-help]
-domain domainName | -net netName
```

Reports the `user_attributes` of the power domain or net specified in the CPF file. If the domain or net has no `user_attributes` in the CPF file, then this command reports an empty string.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `get_cpf_user_attributes` parameter. For a detailed description of the command and all its parameters, use the `man` command: `man get_cpf_user_attributes`
`-domain`	Specifies the name of the power domain containing the user_attributes. *Data_type*: string, optional
`-net`	Specifies the name of the net having the `user_attributes`. *Data_type*: string, optional

Related Topics

The following sections in the Low Power Design chapter of the *Innovus User Guide*:

- Overview
- Power Domain Shutdown and Scaling
- Support for the Common Power Format

get_pg_pins_use_signal_route

```
get_pg_pins_use_signal_route
[-help]
```

Returns all the existing MSV settings that were set by the `set_pg_pins_use_signal_route` command in the current database. Use this command to identify the cell PG pins, which are routed by the signal router.

Parameters

`-help`	Prints out the command usage. For a detailed description of the command and all its parameters, use the `man` command: `man get_pg_pins_use_signal_route`

get_upf_port_attr

```
  get_upf_port_attr
[-help]
hport
```

Returns UPF port attributes from UPF or for the specified values. Run this command before design partitioning. Following is the output format of this command:

```
list of {attr_name val}
```

Parameters

`-help`	Prints a brief description that includes the type and default information for each `get_upf_port_attr` parameter. For a detailed description of the command and all its parameters, use the man command: `man get_upf_port_attr`
hport	Specifies the module term name for which UPF port attributes need to be returned. *Data_type*: string, required

Example

The following command returns UPF port attributes for the `iC/in hport`:

```
get_upf_port_attr iC/in
```

Following is the output of the above command:

```
{-driver_supply supplySetName} {attr_name value2} {is_analog FALSE} …
```

Related Information

Low Power Design chapter in the *Innovus User Guide*

get_valid_power_switch_location

```
get_valid_power_switch_location
-help
-cell cell_name
-loc location
[-orient string]
[-search_area string]
```

Returns a valid location of the power switch cells according to the specified placement grid, cell name, and nearest location. This valid location of the power switch cells can be used to specify offsets while using the `add_power_switches` command.

Parameters

`-help`	Prints a brief description that includes the type and default information for each `get_valid_power_switch_location` parameter. For a detailed description of the command and all its parameters, use the man command: `man get_valid_power_switch_location`
`-cell` *cell_name*	Specifies the name of the power switch cell whose valid location needs to be identified. *Data_type*: string, required
`-loc` *location*	Specifies the nearest location from where the valid location needs to be identified. *Data_type*: string, required
`-orient` *string*	Specifies the orientation. The orientation can be one of the following: `r0`, `r90`, `r180`, `r270`, `mx`, `mx90`, `my`, and `my90`. *Data_type*: string, optional
`-search_area` *string*	Specifies the area within which the valid location needs to be identified. *Data_type*: string, optional

Example

The following command identifies a valid location for the `HSWX1` power switch cell:

```
get_valid_power_switch_location -cell HSWX1 -orient MX -loc {382 970}
```

Following is the output of the above command:

```
382.0350 970.0750 MX
```

Related Topics

`add_power_switches`

identify_physical_power_domains

```
identify_physical_power_domains
[-help]
{-domains physical_domain_list | -reset | -check_only}
[-group_physical_logical_domains {{phyPd lpd1 lpd2 ...} ...}]
```

Marks the physical power domains that need fences in case the power domains share the same primary supply net and the available supply net setting (if specified). The command also enables the logical domains to exist without any fence.

Note: `identify_physical_power_domains` needs to be called again (after the initial use), whenever the tool reads in an power intent file. Else, the file content is not recognized.

Parameters

`-help`	Prints out the command usage. For a detailed description of the command and all its parameters, use the `man` command: `man identify_physical_power_domains`
`-check_only`	Checks the fence sharing against Power/Ground (P/G) nets. *Data_type*: bool, optional
`-domains physical_domain_list`	Marks listed power domains as physical domains (a physical power domain must have a fence). Once a physical domain is marked, the power domain sharing the same P/G nets (not marked in this command) are recognized as logical domain (or fenceless domain). Therefore, this domain can be implemented without a fence. The instances within logical domain are placed within the fence of its corresponding physical domain. This parameter also executes the`-checkOnly` parameter, by default. **Note:** Default power domain is not required to be marked, it is marked by default. This command needs to be called after each run of `commit_power_intent`. Otherwise the physical/logical power domain information will be lost in the database. *Data_type*: string, optional
`-group_physical_logical_domains {{phyPd lpd1 lpd2 ...} ...}`	Groups a list of physical and logical domains for the `identify_physical_power_domains` command. *Data_type*: string, optional
`-reset`	Resets the status marking settings after running `commit_power_intent`. *Data_type*: bool, optional

Example

The following command marks `PDdefault` as physical domain:

```
identify_physical_power_domains -domains {PDdefault}
```

Related Topics

- `commit_power_intent`
- Low Power Design chapter in the *Innovus User Guide*

move_power_switches

```
move_power_switches
[-help]
[-area {x1 y1 x2 y2}]
[-column_x string]
[-fix_overlap]
[-honor_soft_blockage]
[-power_domain string]
[-report_file string]
[-selected]
[-switch_instances string]
[-x_offset float]
[-y_offset float]
```

The command is used to move power switch instances by offset value in X or Y direction. You can specify multiple power switch instances to be moved in the same offset value.

Parameters

`-help`	Prints a brief description that includes the type and default information for each `move_power_switches` parameter. For a detailed description of the command and all its parameters, use the man command: `man move_power_switches`
`-area {x1 y1 x2 y2}`	Specifies moving the switches inside the indicated area. *Data_type*: rect, optional
`-column_x string`	Specifies moving columns at the indicated x locations. *Data_type*: string, optional
`-fix_overlap`	Internally checks any overlap and fixes it with the suggested offset value. *Data_type*: bool, optional
`-honor_soft_blockage`	When used, prevents the `move_power_switches` command from moving power switch cells in areas having soft placement blockage. *Data_type*: bool, optional
`-power_domain string`	Specifies moving the switches found in the indicated power domain. *Data_type*: string, optional
`-report_file string`	Specifies the name of the report file that stores information about the moved switches. *Data_type*: string, optional
`-selected`	Specifies moving the selected switches. *Data_type*: bool, optional
`-switch_instances string`	Specifies moving the switches in the list. *Data_type*: string, optional
`-x_offset float`	Specifies moving switches by the indicated amount in the x direction. *Data_type*: float, optional
`-y_offset float`	Specifies moving switches by the indicated amount in the y direction. *Data_type*: float, optional

Related Topics

- add_power_switches
- check_power_switches
- delete_power_switches
- read_power_switches
- report_power_switches
- update_power_switches
- write_power_switches
- Low Power Design chapter in the *Innovus User Guide*

power_intent Category Attributes

The root attributes in the `power_intent` category are:

- power_intent_allow_power_domain_min_gap_zero
- power_intent_allow_shifter_voltage_mismatch
- power_intent_honor_power_domain_for_domain_crossing_route
- power_intent_honor_power_domain_for_intra_domain_route
- power_intent_do_not_use_top_domain_for_port_voltage
- power_intent_allow_back_to_back_isolation
- power_intent_allow_nested_default_domain
- power_intent_assume_iso_enable_pin_is_always_on
- power_intent_check_all_nets_for_domain_crossing
- power_intent_include_dot_lib_related_pg_pin
- power_intent_share_well_always_on_buffering_support
- power_intent_use_cpf_global_connect_for_always_on_buffer
- power_intent_use_cpf_global_connect_for_shifter
- power_intent_use_effective_domain_for_iso_shifter_insertion

To set the value of any root attribute, use the `set_db` command:

```
set_db attribute_name value
```

To get the current value of any root attribute, use the `get_db` command:

```
get_db attribute_name
```

To get the current value of all attributes in the `power_intent` category, use the following command:

```
get_db -category power_intent
```

power_intent_allow_power_domain_min_gap_zero

```
power_intent_allow_power_domain_min_gap_zero int
```

Default: `true`

Data_type: bool, read/write

Allows users to set power domain zero `minGap`.

power_intent_allow_shifter_voltage_mismatch

```
power_intent_allow_shifter_voltage_mismatch {true | false}
```

Default: `true`

Data_type: bool, read/write

When set to false, it prevents insertion of level shifters by allowing voltage tolerance.

power_intent_honor_power_domain_for_domain_crossing_route

```
power_intent_honor_power_domain_for_domain_crossing_route {true | false}
```

Default: `true`

Data_type: bool, read/write

Routes nets belonging to more than one power domain so that the routing does not violate power domain boundaries, in addition to routing intra-power domain and global nets using the specifications of the power domains.

For example, when you specify `power_intent_honor_power_domain_for_domain_crossing_route true`, if a net belongs to power domains PD1 and PD3, but not PD2, Early Global Route routes the net without going over the PD2 power domain boundary.

power_intent_honor_power_domain_for_intra_domain_route

```
power_intent_honor_power_domain_for_intra_domain_route {true | false}
```

Default: `true`

Data_type: bool, read/write

Routes intra-power domain and global nets using the specifications of the power domains. If you specify `power_intent_honor_power_domain_for_intra_domain_route true`, Early Global Route routes intra-power domain nets only within their power domains. Global nets that do not connect with a power domain do not get routed through that power domain.

power_intent_do_not_use_top_domain_for_port_voltage

```
power_intent_do_not_use_top_domain_for_port_voltage {true | false}
```

Default: `false`

Data_type: bool, read/write

Controls voltage using top fterm domain for IO pin voltage.

power_intent_allow_back_to_back_isolation

```
power_intent_allow_back_to_back_isolation {true | false}
```

Default: `false`

Data_type: bool, read/write

Supports back-to-back isolation insertion along one cross-domain net based on the CPF ISO rule. Set true to enable.

power_intent_allow_nested_default_domain

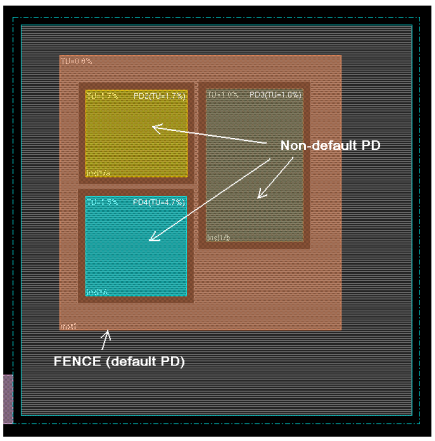
```
power_intent_allow_nested_default_domain {true | false}
```

Default: `true`

Data_type: bool, read/write

Allows a non-default power domain member to be logically nested in a default power domain. This default power domain member has fence constraint.

By default, due to grouping, the fence constraint is lost for the non-default power domain member. Set true to enable.



In the netlist: modules `HInst11` and `HInst12` are under `HInst1`; and `HInst1` is under the top.

You have the following:

```
create_power_domain -instances HInst1

create_power_domain -default -instances {HInst11 HInst12}...
```

That means `HInst11` and `HInst12` are cut from `HInst1` to default power domain. If there is a fence/region constraint for `HInst11` and `HInst12` in default power domain, then please turn this global on. Otherwise, the fence/region for `HInst11` and `HInst12` will be lost.

power_intent_assume_iso_enable_pin_is_always_on

```
power_intent_assume_iso_enable_pin_is_always_on {true | false}
```

Default: `true`

Data_type: bool, read/write

Insert ISO/LS for all the inter-domain nets using effective domain based on the CPF rules. Set `true` to enable.

power_intent_check_all_nets_for_domain_crossing

```
power_intent_check_all_nets_for_domain_crossing {true | false}
```

Default: `true`

Data_type: bool, read/write

Checks all the nets for domain crossings. Set `true` to enable.

power_intent_include_dot_lib_related_pg_pin

```
power_intent_include_dot_lib_related_pg_pin {true | false}
```

Default: `true`

Data_type: bool, read/write

Honors library related PG pins. Set true to enable.

power_intent_share_well_always_on_buffering_support

```
ower_intent_share_well_always_on_buffering_support {true | false}
```

Default: `false`

Data_type: bool, read/write

Shares the isolated Nwell AO buffers.

power_intent_use_cpf_global_connect_for_always_on_buffer

```
power_intent_use_cpf_global_connect_for_always_on_buffer {true | false}
```

Default: `false`

Data_type: bool, read/write

Honors connection pecs for AO buffer based on the CPF rules. Set `true` to enable.

power_intent_use_cpf_global_connect_for_shifter

```
power_intent_use_cpf_global_connect_for_shifter {true | false}
```

Default: `false`

Data_type: bool, read/write

Honors connection specs for shifters based on the CPF rules. Set `true` to enable.

power_intent_use_effective_domain_for_iso_shifter_insertion

```
power_intent_use_effective_domain_for_iso_shifter_insertion {true | false}
```

Default: `true`

Data_type: bool, read/write

Honors effective domains for ISO/LS insertions. Set `true` to enable.

read_power_intent

```
read_power_intent
[-help]
power_intent_file
{-cpf | -1801 | -msv_db }
```

Reads-in power intent file and do syntax check. The method supports incremental refinement of power intent specifications required for IP-based design flows.

Note: It is needed to read the power intent using `read_power_intent [-cpf|-1801] file` before running the `init_design` command for design initialization.

Parameters

`-help`	Prints a brief description that includes the type and default information for each `read_power_intent` parameter. For a detailed description of the command and all its parameters, use the `man` command: `man read_power_intent`
power_intent_file	Specifies the name of the power intent file to be read. *Data_type*: string, required
`-cpf`	Specifies that the CPF format power intent file will be used. *Data_type*: bool, required
`-1801`	Specifies that the IEEE1801 format power intent file will be used. *Data_type*: bool, required
`-msv_db`	Specifies that the IEEE1801 data format `(.cpfdb)` power intent file will be used. **Note:** `-msv_db` is for partition flow and must be read for power intent in partition implementation. *Data_type*: bool, required

Related Topics

- `commit_power_intent`
- `delete_power_intent`
- `init_design`
- `write_power_intent`
- Low Power Design chapter in the *Innovus User Guide*

read_power_switches

```
read_power_switches
[-help]
[-power_domain powerDomainName]
[-in_file fileName]
```

Imports power switch databases. The need for this commands arises in case of defIn-based flows where power switch database required for power switch related operations is not loaded through `defOut`.

So, using `write_power_switches` a user can save power switch database after power switch insertion using `add_power_switches` for later use and it can be loaded back using `read_power_switches`.

Note: The enable net IN and OUT connections should remain the same in the netlist in power switch database export/import process and the commands need not be run when `save_design` is run because the power switch database is automatically saved and restored.

PSO File Generation and Usage

A PSO file gets generated as an output of the `write_power_switches` command, which can be used in a future session, if needed. One scenario where this file can be needed is when a DEF file containing a power switch chain is not used in the power switch synthesis and hence the incoming netlist to Innovus would not have any power switches. The power switch chain is then created at the `init` stage of Innovus using the `add_power_switches` command in such a run. A database saved using `write_db` after running the `add_power_switches` command, will have the power switch chain saved in a `<design.pso>` file. When restored, this database can also be used to write out the saved PSO file using the `write_power_switches` command. When the power switches are added using the `add_power_switches` command, the `report_power_switches` command can be used to validate their addition.

To avoid regenerating power switches in future implementation runs, save the power switch chain using the `write_power_switches` command along with the floorplan DEF. Ensure that the `-routing` parameter of the `write_def` command is also used along with `-floorplan` and `-no_std_cells` to save the power switch network under the NETS section of the DEF.

Refer to the following flows to understand the usage of the PSO file:

Innovus Session 1:

```
<init Design1 - incoming design has no power switches>
add_power_switches
write_power_switches test.pso
write_def -floorplan -no_std_cells -routing test.def
```

Innovus Session 2:

```
<init Design2 - incoming design has no power switches>
read_power_switches -in_file test.pso
read_def test.def
```

While reading the DEF file, you can identify the power switches using the database commands, even without using the `read_power_switches` command. Following is a usage example of such a database command:

```
get_db [get_db insts -if {.name == inst1/psoI_PD1_1_HSWX1_200_1582_0} ] .base_cell.is_power_switch
```

Here, `inst1/psoI_PD1_1_HSWX1_200_1582_0` is the instance name of the switch cell.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `read_power_switches` parameter. For a detailed description of the command and all its parameters, use the `man` command: `man read_power_switches`
`-power_domain` *`powerDomainName`*	Specifies the power domain whose power switch information is to be imported. *Data_type*: string, optional
`-in_file` *`fileName`*	Specifies the name of the import file. *Data_type*: string, optional

Related Topics

- `add_power_switches`
- `check_power_switches`
- `delete_power_switches`
- `move_power_switches`
- `report_power_switches`
- `update_power_switches`
- `write_power_switches`
- Low Power Design chapter in the *Innovus User Guide*

read_power_switches_for_floorplan

```
read_power_switches_for_floorplan
[-help]
[-file string]
```

Restores the information stored in the power switch databases. You can use this command to load back the power switch information only after completing power switch insertion using the `add_power_switches` command and then saving the database information using the `write_power_switches_for_floorplan` command.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `read_power_switches_for_floorplan` parameter. For a detailed description of the command and all its parameters, use the `man` command: `man read_power_switches_for_floorplan`
`-file string`	Specifies the name of the file from which the power switch database information will be restored.

Related Information

- `add_power_switches`
- `write_power_switches_for_floorplan`

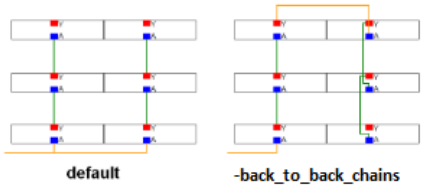
reorder_power_switch_chains

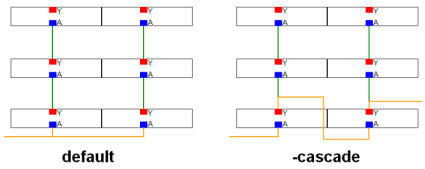
```
reorder_power_switch_chains
[-help]
[-back_to_back_chains string]
[-cascade]
[-cell_enable_pins string]
[-chain_by_instances]
[-chain_by_rows]
[-chain_direction_x LtoR|RtoL]
[-chain_direction_y BtoT|TtoB]
[-chain_style {min_length_tree single_output_chain}]
[-chain_x_before_y]
[-chain_y_before_x]
[-enable_nets_in {listOfnets}]
[-enable_nets_out  {netBaseName | {listOfNets}}]
[-enable_pins_in  {listOfPins}]
[-enable_pins_out  {listOfPins}]
[-max_distance_x um]
[-max_distance_y um]
[-merge_distance_x um]
[-merge_distance_y um]
[-parallel_enable]
[-out_file string]
[-reverse_order]
[-selected]
[-switch_insts string]
[-unchain_by_insts]
```

Enables flexible power switch enable chaining and unchaining after power switch insertion.

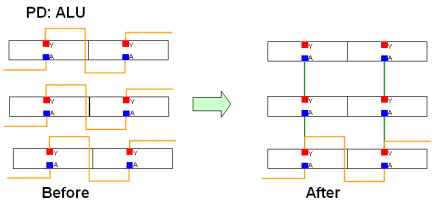
Note: It is recommended that you use this command for PSO enable chaining.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `reorder_power_switch_chains` parameter. For a detailed description of the command and all its parameters, use the `man` command: `man reorder_power_switch_chains`
`-back_to_back_chains` *string*	Connects the `enable_nets_out` at the top of a column to the `enable_nets_in` at the top of the next column, and connects the `enable_nets_out` at the bottom of the column to the `enable_nets_in` at the bottom of the next column. The chain proceeds left to right or right to left. By default, the `enable_nets_in` is connected to the bottom of each column, and the `enable_nets_out` exits from the top of each column, in parallel.  This parameter is optional. *Data_type*: string, optional

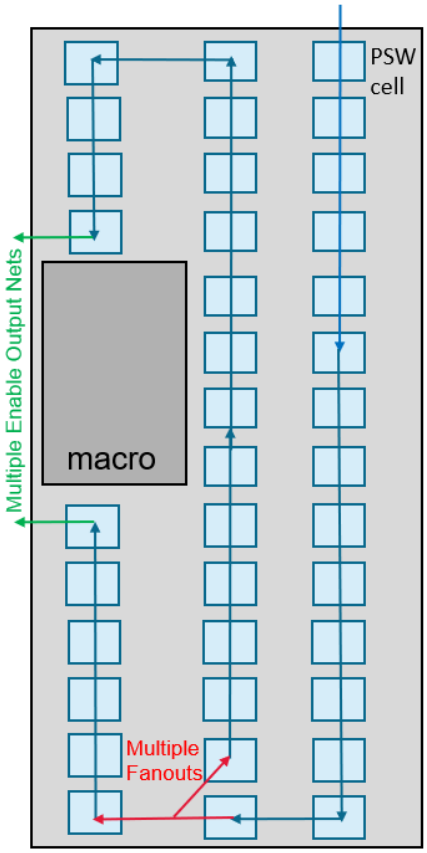
`-cascade`	Connects switches in cascade  *Data_type*: bool, optional

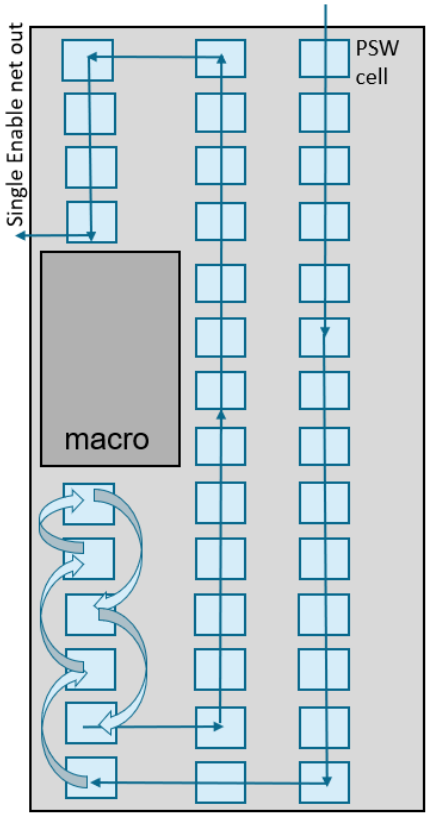
`-cell_enable_pins` *string*	Provides the pin names of power switch cells when the power switch cell pin names are different than the specified instance's cell. This option is also useful in cases where all enable pins of instances are not connected by enabling a way to provide pin map.
	Example:
	`set INST [ add_power_switches -get_switch_instances -power_domain PD1 -column ]`
	`reorder_power_switch_chains -chain_by_instances -chain_y_before_x \`
	`-unchain_by_insts -enable_pins_in SW_IN -enable_pins_out SW_OUT \`
	`-enable_nets_in NETin -enable_nets_out NETout \`
	`-chain_direction_x RToL -out_file pd1_chain.rpt \`
	`-back_to_back_chains \`
	`-cell_enable_pins { { SW_withDiff_pinNames {SW_IN_1} {SW_OUT_1} }} \`
	`-switch_insts $INST`
	Data_type: string, optional

`-chain_by_instances`	To chain the instances. This option is used for chaining the instances defined by `-selected` or instances given in `-switch_insts`.
	 
	
	Select all PSO instances of power domain ALU
	`set ALUinst [add_power_switches -column -get_switch_instances -power_domain ALU]`
	The below command make the changes in the connection of the switch enable
	`reorder_power_switch_chains -unchain_by_insts\`
	`-chain_by_instances \`
	`-switch_insts $ALUinst -enable_pins_in A -enable_pins_out Y \`
	`-enable_nets_in swcontrol_2 -enable_nets_out swack_2 \`
	`-cascade -chain_direction_y BtoT`
	Data_type: bool, optional

`-chain_by_rows`	Gives additional control if you need to enable horizontal signal chaining. For example, in rows. By default, signal chaining is enabled from bottom to top. *Data_type*: bool, optional

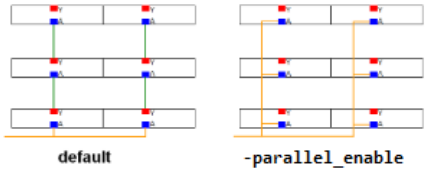
`-chain_direction_x` *`LtoR|RtoL`*	Defines the direction of the chain along X axis. *Data_type*: enum, optional
`-chain_direction_y` *`BtoT|TtoB`*	Defines the direction of the chain along Y axis. *Data_type*: enum, optional

`-chain_style {min_length_tree single_output_chain}`	Specifies the chain style for enabling power switches and making enable chain connections. The specified chain styles help in optimizing the wire length with longer enable chains, and connecting all the power switches in single pass to avoid blockages. Following are the chain style values specified by this parameter:
	• `min_length_tree`: Connects the enable pin of the switches with a chain, which has multiple branch outputs. The following command presents the usage of this parameter value:
	`reorder_power_switch_chains -power_domain PD1 -selected \`
	`-chain_style min_length_tree \`
	`-max_distance_x 40 -max_distance_y 100 -start_loc "200 6400" \`
	`-enable_nets_in pso1 -enable_nets_out pso1_out ….`
	 
	
	• `single_output_chain`: Connects the enable pin of the switches with a chain, which has single branch output. This parameter value can be specified with `-max_distance_x` and `-max_distance_y` parameters. The following command presents the usage of this parameter value:
	`reorder_power_switch_chains -power_domain PD1 -selected \`
	`-chain_style single_output_chain \`
	`-max_distance_x 40 -max_distance_y 100 -start_loc "200 6400" \`
	`-enable_nets_in pso1 -enable_nets_out pso1_out ….`



Data_type: enum, optional

`-chain_x_before_y`	Defines row chain over column. *Data_type*: bool, optional
`-chain_y_before_x`	Defines column chain over row. This is the default option. *Data_type*: bool, optional
`-enable_nets_in {listOfnets}`	Specifies the list of input nets that the software attaches to the enable input pins. The order in which you specify these nets determines the sequence in which nets are assigned to pins. Enable nets and pins are connected automatically if they reside at the same level in the hierarchy. *Data_type*: string, optional
`-enable_nets_out {`*`netBaseName`*` | {listOfNets}}`	Specifies the enable output nets that the software attaches to the enable output pins. The order in which you specify these nets determines the sequence in which nets are assigned to pins. If enable nets and pins reside at the same level in the hierarchy they are connected automatically. For the last switch of the ring, the `-enable_pins_out` is tied to `-enable_nets_out`. *Data_type*: string, optional
`-enable_pins_in {listOfPins}`	Specifies the list of enable input pins of the buffers inside the switch cell. The order in which you specify these nets determines the sequence in which nets are assigned to pins. If enable nets and pins reside at the same level in the hierarchy they are connected automatically.The software ties the `-enable_pins_out` of the first switch cell to the `-enable_pins_in` of the next switch cell in the same ring: All the enable pin of the switch cells in the same column are daisy-chained together. For the first switch of the ring, the `-enable_pins_in` is tied to `-enable_nets_in`. *Data_type*: string, optional

`-enable_pins_out {listOfPins}`	Specifies the list of enable output pin(s) of the buffer(s) inside the switch cell. The order in which you specify these pins is important because nets are assigned to pins sequentially. The software ties the `-enable_pins_out` of the first switch cell to the `-enable_pins_in` of the next switch cell in the same column: All the enable pin of the switch cells in the same column are daisy-chained together. For the last switch of the column, the `-enable_pins_out` is tied to `-enable_nets_out`. *Data_type*: string, optional
`-max_distance_x` *um*	Maximum X distance a switch can drive. *Data_type*: float, optional
`-max_distance_y` *um*	Maximum Y distance a switch can drive. *Data_type*: float, optional
`-merge_distance_x` *um*	To consider switches within this in the same column. *Data_type*: float, optional
`-merge_distance_y` *um*	To consider switches within this in the same row. *Data_type*: float, optional
`-parallel_enable`	Connects enable Net In to all switches in parallel type connection.  *Data_type*: bool, optional
`-out_file` *string*	Writes switch instances and their corresponding input/output enable signal pin/net connections to the specified file name. *Data_type*: string, optional
`-reverse_order`	The reverse order of the `-switch_insts`. *Data_type*: bool, optional
`-selected`	Selects switches to be chained and unchained. **Note:** This option must be used with `-chain_by_instances` and `-unchain_by_insts`. *Data_type*: bool, optional
`-switch_insts` *string*	Specifies the instances to be chained/unchained. When chaining is requested, the sequence of the final chain is taken from the sequence of the instance names given in the provided list. *Data_type*: string, optional
`-unchain_by_insts`	To unchain the instances. This option is used for unchaining the instances defined by `-selected` or instances given in `-switch_insts`. *Data_type*: bool, optional

report_isolation

```
report_isolation
[-help]
[-highlight]
[-out_file fileName]
[-from powerDomain]
[-to powerDomain]
```

Reports or shows the added isolation cells and instances.

Note: Use this command after you add isolation cells.

Parameters

-help	Outputs a brief description that includes the type and default information for each `report_isolation` parameter. For a detailed description of the command and all its parameters, use the `man` command: `man report_isolation`
`-from`	Reports all the isolation instances connected to the output pins of the power domain specified in `-from` option. *Default:* Reports all isolation cells. *Data_type*: string, optional
`-highlight`	Highlights isolation instances in the main window. *Data_type*: bool, optional
`-out_file` *fileName*	Writes isolation cell and instance names to a report. *Default:* The software writes the report to a default file: *designName*`.isolation.rpt` *Data_type*: string, optional
`-to`	Reports all the isolation instances connected to the input pins of the power domain specified in `-to` option. *Default:* Reports all isolation cells. *Data_type*: string, optional

Example

The following command highlights isolation cells and writes the cell names to `isolationCellReport`:

```
report_isolation -highlight -out_file isolationCellReport
```

Related Topics

Low Power Design chapter in the *Innovus User Guide*

report_partition_boundary_ports

```
report_partition_boundary_ports
-pins
```

The `report_partition_boundary_ports` command prints out the boundary pins' debug information during partition (through `commit_partitions`) in a log which shows how to assign the power domain for the partition pins.

This is useful when debugging CPF partitions as you get the information about how Innovus assigns the power domain for the partition boundary pins during debugging.

Parameters

-pins	Specifies a list of the partition pins. If wildcard * is specified, it means all partition pins. *Data_type*: string, required

Example

Specify the partition pins(Hterm) you want to debug:

```
report_partition_boundary_ports "I2/d I2/clk"
```

After partition, the command will execute the partition command to print the following DBG info in the log file:
DBG: deriving hterm I2/d power domain
DBG: ….

Related Topics

Low Power Design chapter in the *Innovus User Guide*

report_power_domains

```
report_power_domains
[-help]
[-bind_lib]
[-insts inst_name]
[-iso_inst]
[-modules {list_of_modules}
[-nets net_name]
[-out_file outFile]
[-partition_hports pin | port | hpin+]
[-pg_nets]
[-pins]
[-power_domain powerDomainName [-check_pg_scope]]
[-power_domain_boundary_port string]
[-power_domain_coverage pd1 pd2]
[-shifter]
[-supply_crossing supp1 supp2]
[-verbose]
[-voltage]
```

Reports information about power domains. The report includes the following attributes:

- Power domain name
- Power domain bounding box
- Timing library
- Mingap
- RS extension
- Row type
- Row spacing

The command also reports whether the power domain is always on.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `report_power_domains` parameter. For a detailed description of the command and all its parameters, use the `man` command: `man report_power_domains`
`-bind_lib`	Reports the timing library bound to each instance of a power domain. *Data_type*: bool, optional
`-check_pg_scope`	Checks if there are any PG connections that cross UPF-defined scope. All PG connections for UPF scope should be at the boundary PG port. This parameter is used with the `-power_domain` parameter. *Data_type*: bool, optional

`-insts inst_name`	Reports the power domain-related information for the specified instance: the input, output, and power/ground connections. *Data_type*: string, optional
`-iso_inst`	Reports all isolation cells in the power domain. *Data_type*: bool, optional
`-modules {list of modules}`	Checks whether the specified instances or hierarchical instances are in the power domain. *Default:* The software does not perform this check. *Data_type*: string, optional
`-nets net_name`	Reports power domain-related information for the specified net: • The driver instance of the net and its power domain. • The receiver instances of the net and their power domains. *Data_type*: string, optional
`-out_file outFile`	Specifies the name of the output file. *Default:* `designName-powerdomain.rpt` *Data_type*: string, optional
`-partition_hports pin | port | hpin+`	Reports the hierarchical partition (Hinst) boundary port's domain at the time of partitioning. *Data_type*: (pin|port|hpin)*, optional
`-pg_nets`	Reports all the power/ground nets in the power domain. *Data_type*: bool, optional
`-pins pin1, pin2 ...`	Lists the pins and the option is of type string. This option is optional. *Data_type*: string, optional
`-power_domain powerDomainName`	Specifies the power domain whose attributes you want to report. *Default:* If you do not specify the power domain, this command report data about all power domains. *Data_type*: string, optional
`-power_domain_boundary_port string`	Reports if the specified port name of the power domain boundary has an IEEE1801 constraint. *Data_type*: string, optional

`-power_domain_coverage` *pd1 pd2*	Reports the coverage between two specified domains. *Data_type*: string, optional
`-shifter`	Reports all shifters in the power domain. *Data_type*: bool, optional
`-supply_crossing` *supp1 supp2*	Reports if the two specified supplies have the on/off and low/high crossing. *Data_type*: string, optional
`-verbose`	Prints out the detail information and is optional. *Data_type*: bool, optional
`-voltage`	Reports domain voltage. *Data_type*: bool, optional

Examples

- The following command checks the `I1/PD1` instances to identify if there are any PG connections that do not use the `UPF I1` PG nets. If those are found, an error occurs specifying the instance name and its out-of-scope PG connection.
  ```
  report_power_domains -check_pg_scope true -power_domain I1/PD1
  ```
- The following command reports all shifter cells, all isolation cells, and all power/ground nets in power domain `PD1`. The command also checks whether HInst `instA` is contained in `PD1`, and reports all results to `myDesign-powerdomain.rpt`.
  ```
  report_power_domains -power_domain PD1 -out_file myDesign-powerdomain.rpt -modules {instA} -shifter -iso_inst -pg_nets
  ```
- The following example shows how output file `outfile` contains the binding between the timing libraries and instances in power domain VCORE:
  ```
  report_power_domains -bind_lib -power_domain VCORE -out_file outfile

  Timing library for instance(s):
  ============================
  INST = uLSCVSocDownVCore/uACLKEND/uLVHL (lvl_slow_1v08_1v08)
  INST = uLSCVSocDownVCore/uACLKENI/uLVLHL (lvl_slow_1v08_1v08)
  INST = uLSCVSocDownvCore/uaCLKENP/uLVLHL (lvl_slow_1v08_1v08)
  INST = uLSCVSocDownvCore/uaCLKENRW/uLVLHL (lvl_slow_1v08_1v08)
  INST = uLSCVSocDownvCore/uaBIGENDINIT/uLVHlL (lvl_slow_1v08_1v08)
  INST = uLSCVSocDownvCore/uaCLKIN/uLVHL (lvl_slow_1v08_1v08)
  INST = uLSCVSocDownvCore/uCP15SDISABLE/uLVHL (lvl_slow_1v08_1v08)
  ```
- The following command reports the power domain-related information about instance `PM_INST/state_inst/g45`:
  ```
  report_power_domains -insts PM_INST/state_inst/g45
  ```

  Result:

  ```
  Analyzing PM_INST/state_inst/g45 (Cell: AN2D1) power domain:AO
  ------from side ------
  Pin A2 => Net PM_INST/state_inst/n_28
  PM_INST/state_inst/g71/ZN in power domain A0
  Pin A1 => Net PM_INST/state_inst/n_25
  PM_INST/state_inst/g47/Z in power domain A0
  ```

```
------to side ------
Pin Z => Net PM_INST/state_inst/n_36
PM_INST/state_inst/fsm_reg[1]/qi_reg/D in power domain A0
==========PG pins =============
VDD conneted to Net: VDD
VSS connected to Net:VSS
```

Related Topics

- The following sections in the Low Power Design chapter of the *Innovus User Guide*:
  - Overview
  - Power Domain Shutdown and Scaling
  - Multiple Supply Voltage Flat Flow

report_power_switches

```
report_power_switches
[-help]
-out_file file_name
```

Reports each power switch instance, the location, and pin/net connections. The report format is as follows:

```
#InstName (x,y)
#   (pin1): net1
#   (pin2): net2
```

Parameters

`-help`	Outputs a brief description that includes the type and default information for each report_power_switches parameter. For a detailed description of the command and all its parameters, use the `man` command: `man report_power_switches`
`-out_file file_name`	Specifies the name of the report file where the information is written. *Data_type*: string, required

Related Topics

- `add_power_switches`
- `check_power_switches`
- `delete_power_switches`
- `move_power_switches`
- `read_power_switches`
- `update_power_switches`
- `write_power_switches`
- Low Power Design chapter in the *Innovus User Guide*

report_shifters

```
report_shifters
[-help]
[-cell cellName]
[-from powerDomainName]
[-to powerDomainName]
[-out_file fileName]
[-highlight]
```

Reports the level shifters that have been read into or defined for the Innovus software. In CPF flow, the level shifters are inserted.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `report_shifters` parameter. For a detailed description of the command and all its parameters, use the man command: `man report_shifters`
`-cell`	Reports shifters with the specified cell. *Default:* Reports all level shifters. *Data_type*: string, optional
`-from powerDomainName`	Reports all the shifter instances connected to the output pins of the power domain specified in the `-from` option. *Default:* Reports all level shifters. *Data_type*: string, optional
`-highlight`	Highlights placed level shifters. *Data_type*: bool, optional
`-out_file fileName`	Reports all instances in an output file. *Default:* The instances will be reported to a default output file `designName.shifter.rpt`. A warning message is printed to indicate a default output file is created. *Data_type*: string, optional
`-to powerDomainName`	Reports shifter instances connected to the input pins of the power domain specified in the `-to` option. *Default:* Reports all level shifters. *Data_type*: string, optional

Related Topics

The following sections in the Low Power Design chapter of the *Innovus User Guide*:

- Overview

- Power Domain Shutdown and Scaling

report_voltage

```
report_voltage
[-help]
[-early]
[-inst string]
[-late]
[-net string]
[-pin string]
[-quiet]
[-view string]
```

This command reports the voltage for the net/pin (term)/inst in the specific view under a specific corner.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `report_voltage` parameter. For a detailed description of the command and all its parameters, use the man command: `man report_voltage`
`-early`	Specifies that the voltage is reported in the early mode. *Data_type*: bool, optional
`-inst string`	Specifies the instance power domain voltage (PVT). *Data_type*: string, optional
`-late`	Specifies that the voltage is reported in the late mode. *Data_type*: bool, optional
`-net string`	Specifies the net voltage (driver's pin voltage). *Data_type*: string, optional
`-pin string`	Specifies the term voltage. *Data_type*: string, optional
`-quiet`	Displays only the pin voltage-related information without other additional unnecessary information to avoid huge log size. *Data_type*: bool, optional
`-view string`	Specifies the analysis view. *Data_type*: string, optional

Related Topics

Low Power Design chapter of the *Innovus User Guide* document

route_secondary_pg_pins

```
route_secondary_pg_pins
[help]
[-all]
[-max_pins_per_trunk number]
[-nets {listofNets}]
[-non_default_rule ruleName]
[-pattern {trunk | steiner}]
[-well]
```

Routes the secondary power/ground pins of always-on cells to the specified power/ground nets in signal route. You can also control the routing pattern and wire by specifying the pattern, the maximum number of pins per cluster, and non-default rules.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `route_secondary_pg_pins` parameter. For a detailed description of the command and all its parameters, use the man command: `man route_secondary_pg_pins`
`-all`	Routes non-primary power/ground pins of all the cells. *Data_type*: bool, optional
`-max_pins_per_trunk` *number*	Specifies the maximum number of pins per cluster in the trunk rotating pattern. *Default:* 5 *Data_type*: int, optional
`-nets` *listOfNets*	Specifies the power/ground nets to be connected to the secondary power/ground pins. *Data_type*: string, optional
`-non_default_rule` *ruleName*	Specifies the non-default rule the signal router uses to route the secondary power/ground pins. *Default:* Regular signal routing rule is used by default. *Data_type*: string, optional
`-pattern {trunk | steiner}`	Specifies how the signal router routes the secondary power/ground pins. `Trunk`: When the `max_pins_per_trunk` option's value is 1, it routes the pins to nearest power grid. When the value is more than 1, routes all pins per cluster to the trunk. `Steiner`: Routes all pins to a trunk. *Default:* `trunk` *Data_type*: enum, optional
`-well`	Includes nwell and pwell pins. *Data_type*: bool, optional

Related Topics

The following sections in the Low Power Design chapter of the *Innovus User Guide*:

- Overview
- Power Domain Shutdown and Scaling

set_pg_pins_use_signal_route

```
  set_pg_pins_use_signal_route
[-help]
[-reset]
[-all]
{cell1:pin1 [cell2:pin2 ...]} cellName:pinName cellName:pinName2 ...
[-well]
```

Sets the internal bit for a given cell/pin pair to be routed by the signal router. Use this command to enable NanoRoute to route always-on-pins in SRPG (state retention power gating) or always-on buffer cells.

Parameters

`-help`	Prints out the command usage.
`-all`	Routes all backup and non-main_rail primary power/ground pins. *Data_type*: bool, optional
`{cell1:pin1 [cell2:pin2 ...]}` *cellName*	Specifies the cell containing the pin for NanoRoute to route. You can use a wildcard to specify all cells of certain type. *Data_type*: list, optional
`-reset`	Resets this feature. *Data_type*: bool, optional
`{cell1:pin1 [cell2:pin2 ...]}` *pinName*	Specifies the pin to be routed. *Data_type*: list, optional
`-well`	Includes nwell and pwell pins. *Data_type*: bool, optional

Examples

- The following example sets both the `vddo` and `vsso` pins for the signal router, but NanoRoute will only route `vddo`:

  ```
  set_pg_pins_use_signal_route SDRFF:vddo SDRFF:vsso
  set_pg_pins_use_signal_route -nets VDD
  ```

- The following example sets NanoRoute to route the `VDD1` pin of all `NAND` cells:

  ```
  set_pg_pins_use_signal_route NAND*:VDD1
  ```

swap_cell_with_always_on_cell

```
swap_cell_with_always_on_cell
[-help]
[-cell_map_file file_name]
[-default_always_on_cell ao_cell_names]
[-nets lp_error_net_list]
[-verbose]
```

Identifies the regular buffer/inverter and swaps it with always-on buffer/inverter. The regular buffer/inverter causes the LP error. Hence, this command fixes the LP error through the regular to always-on buffer/inverter swapping.

Parameters

`-help`	Prints a brief description that includes the type and default information for each `swap_cell_with_always_on_cell` parameter. For a detailed description of the command and all its parameters, use the man command: `man swap_cell_with_always_on_cell`
`-cell_map_file` *file_name*	Specifies the map file name. The map file provides the equivalent driving strength between the regular buffer/inverter and AO buffer/inverter. This means that the timing will not be degraded with using this parameter, when the mapped always-on cell is swapped in. Following is the map file format: `regularBufCellName AoBufCellName` *Data_type*: string, optional
`-default_always_on_cell` *ao_cell_names*	Specifies the default always-on buffer/inverter to be used to replace the regular buffer/inverter. When using default always-on buffer/inverter, this parameter just corrects the LP error. Additionally, this parameter may degrade the timing, and you need to run optimization in flow. *Data_type*: string, optional
`-nets` *lp_error_net_list*	Specifies the list of LP-error nets. By default, the `swap_cell_with_always_on_cell` command works on all the buffer/inverter trees. When this parameter is used, the command works on the buffer/inverter tree for the specified nets. *Data_type*: string, optional
`-verbose`	Displays the detailed trace information for the specified nets. *Data_type*: bool, optional

Examples

- The following command swaps the incorrect regular buffer with `AoBuf1`, and the incorrect regular inverter with `AoInverter1`:
  `swap_cell_with_always_on_cell -default_always_on_cell "AoBuf1 AoInverter1"`
- The following command swaps the incorrect regular buffer/inverter using the specified mapping cells or the default always-on cells:
  `swap_cell_with_always_on_cell -cellMapFile mapfile1 -default_always_on_cell "AoBuf1 AoInverter1"`
- The following command swaps the incorrect regular buffer/inverter using the default always-on cells for the buffer/inverter trees along with the `net1` and `net2` paths:
  `swap_cell_with_always_on_cell -nets "net1 net2" -default_always_on_cell "AoBuf1 AoInverter1"`
- The following command swaps the incorrect buffer/inverter along with the `net1` and `net2` paths and displays the detailed trace information:
  `swap_cell_with_always_on_cell -nets "net1 net2" -default_always_on_cell "AoBuf1 AoInverter1 -verbose"`

update_power_domain

```
update_power_domain
[-help]
[-add_block_box string]
[-core_to_bottom valueInMicron]
[-core_to_left valueInMicron]
[-core_to_right valueInMicron]
[-core_to_top valueInMicron]
[-default_tech_site string]
[-disjoint_hinst_box_list string]
[-first_row_site_index integer]
[-gap_sides {values}|-gap_edges {values}]
[-last_row_site_index integer]
power_domain_name
[-power_extend_sides {values}|-power_extend_edges {values}]
[-row_flip {first second noflip auto}]
[-row_pattern_site site_name]
[-row_space_type {0 1 2}]
[-row_spacing float]
```

Renames a power domain or changes its structure.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `update_power_domain` parameter. For a detailed description of the command and all its parameters, use the `man` command: `man update_power_domain`
`-add_block_box`	Adds a hard block to the disjoint power domain. `-add_block_box` supports both rectangular and rectilinear macros. *Data_type*: string, optional
`-core_to_bottom valueInMicron`	Specifies the row margin from bottom boundary. *Data_type*: float, optional
`-core_to_left valueInMicron`	Specifies the row margin from left boundary. *Data_type*: float, optional
`-core_to_right valueInMicron`	Specifies the row margin from right boundary. *Data_type*: float, optional
`-core_to_top valueInMicron`	Specifies the row margin from top boundary. *Data_type*: float, optional
`-default_tech_site string`	Allows selection of one of the tech sites to be not picked by the software, in case there are two smallest tech sites of the same size in a power domain. **Note:** The specified `-default_tech_site` size value must be same as the current default tech site in the power domain.

`-disjoint_hinst_box_list`	Specifies a region for an Hinst. It is defined as an Hinst and Region pair. *Data_type*: string, optional
`-first_row_site_index` *integer*	Specifies the first row site index value. *Data_type*: int, optional
`-gap_edges`	Specifies the minimum space value to other domain by power domain edge in microns. You can specify the value per edge in clockwise order, starting with the vertical edge at the lower-left corner (smallest Y, and then smallest X). *Data_type*: string, optional
`-gap_sides` *T B L R*	Defines the distance, in microns, that must be reserved from the power domain boundary edges for power routing. You can specify four values, one for each edge. *Data_type*: string, optional
`-last_row_site_index` *integer*	Specifies the last row site index value. *Data_type*: int, optional
power_domain_name	Specifies the current name of the power domain to be modified. This is a positional parameter, and you must specify this value immediately after the command name. *Data_type*: (power_domain), required
`-power_extend_edges`	Specifies the maximum search distance for power extension connections by domain edge in microns. You can specify the value per edge in clockwise order, starting with the vertical edge at the lower-left corner (smallest Y, and then smallest X). *Data_type*: string, optional
`-power_extend_sides` *T B L R*	Specifies the boundary for legal targets to be used by the power planning and routing commands, in conjunction with the power domain boundary. You can specify four values, one for each edge. *Data_type*: string, optional

`-row_flip {first | second | noFlip | auto}`	Flips the orientation of either the first or second row, then follows the flipped and abutted row pattern.
	`first`
	(from bottom to top). MX -> R0 -> MX -> R0 - > …
	`second`
	(from bottom to top). R0 -> MX -> R0 - > MX -> …
	`noFlip`
	(from bottom to top). R0 -> R0 -> R0 - > R0 -> …
	The noFlip option preserves the R0 orientation for all rows (above).
	`auto`
	Detects orientation of row outside of power domain and aligns with power domain bottom. Orientation is applied to the first row at the bottom of power domain.
	So if row on left is R0, the pattern will be (from bottom to top). R0 -> MX -> R0 - > MX -> …
	If row on left is MX, the pattern will be (from bottom to top). MX -> R0 -> MX -> R0 - > …
	The `auto` option resets the software for this automatic behavior if you have previously specified `first`, `second`, or `noFlip`.
	Default: `auto`
	Note: *saveRowImage* resets the option `-row_flip` to `auto`. In this case, try moving the power domain and check the row changes.
	Data_type: enum, optional

`-row_pattern_site site_name`	Specifies the site name for the row pattern. *Data_type*: string, optional
`-row_space_type {0|1|2}`	Determines whether the row spacing value applies between no row (0), each row (1), or each pair of rows (2). *Data_type*: enum, optional
`-row_spacing value`	Specifies the row spacing between each row (1) or each pair of rows (2) as specified in `-row_space_type`. *Data_type*: float, optional

Examples

The following command changes the route search extension values of `PowerDomain1` to 15 µm on each side:

```
update_power_domain PowerDomain1 -power_extend_sides 15 15 15 15
```

Related Topics

Low Power Design chapter in the *Innovus User Guide*

update_power_switches

```
update_power_switches
[-help]
-insts inst_name_list
-cell cell_name
[-x_offset value_in_microns]
[-y_offset value_in_microns]
[-orient orienation_syntax]
[-cell_enable_pins {{cellname {enableInputPinList} {enableOutputPinList}}}]
[-x_y_range_from_center_inst]
```

Replaces the cell of a power switch instance. You can also change the orientation of the instance after cell substitution, and provide a new location for the instance origin relative to the original instance origin.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `update_power_switches` parameter. For a detailed description of the command and all its parameters, use the `man` command: `man update_power_switches`
`-cell` *`cell_name`*	Specifies the cell to substitute for the instance's cell. *Data_type*: string, required
`-cell_enable_pins` {{*`cellname`* {*`enableInputPinList`*} {*`enableOutputPinList`*}}}	Provides the pin names of power switch cell specified thru `-cell` option in `update_power_switches` when the power switch cell pin names are different than the specified instance's cell. This option is also useful in cases where all enable pins of instances are not connected by enabling a way to provide pin map. *Data_type*: string, optional
`-insts` *`inst_name_list`*	Specifies the instance whose cell you want to replace. *Data_type*: string, required
`-orient` *`orientation_syntax`*	Specifies the orientation of the instance after the cell substitution. Choose one of the following values: `r0`, `r90`, `r180`, `r270`, `mx`, `my`, `mx90`, and `my90`. *Default:* The same orientation as the original instance. *Data_type*: enum, optional
`-x_offset` *`value_in_microns`*	Specifies the instance placement in the `x` direction after the cell substitution. A positive value moves the instance to the right of the original instance origin. A negative value moves the instance to the left of the original instance origin. *Data_type*: float, optional

`-x_y_range_from_center_inst`	Selects the power switch instances whose lower left corner is in the range from lower left corner of the specified instance (`-insts`) for replacement with switch cell (`-cell`). **Note:** The first instance specified in the `-insts` will be used for the area computation. The rest of the instances in the `-insts` list will be ignored and will not be replaced. For Example: `update_power_switches -insts psoI_TDSP2_1_CDN_COLUMN_SW_669_1622_3 \ -x_y_range_from_center_inst { 60.3 20 } -cell \ CDN_COLUMN_SW` *Data_type*: string, optional
`-y_offset` *value_in_microns*	Specifies the instance placement in the y direction after the cell substitution. A positive value moves the instance above the original instance origin. A negative value moves the instance above the original instance origin. *Data_type*: float, optional

Examples

- The following example substitutes cell `CDN_RING_SW1` for the original cell for instance `pso1_TDSP2_1_CDN_RING_SW_636_1309_6`:
  `replacePowerSwtich -insts pso1_TDSP2_1_CDN_RING_SW_636_1309_6 -cell CDN_RING_SW1`
- The following example substitutes cell `CDN_RING_SW1` for the original cell for instance `pso1_TDSP2_1_CDN_RING_SW_636_1315_8` and changes the orientation of the instance to `my90`:
  `update_power_switches -insts pso1_TDSP2_1_CDN_RING_SW_636_1315_8 -cell CDN_RING_SW1 -orient my90`

Related Topics

- `add_power_switches`
- `check_power_switches`
- `delete_power_switches`
- `move_power_switches`
- `read_power_switches`
- `report_power_switches`
- `write_power_switches`
- Low Power Design chapter in the *Innovus User Guide*

write_power_intent

```
write_power_intent
[-help]
file_name
-incremental_1801
{-cpf [[-hinsts -top_hier] | [-hinst_only]] | -1801}
[-no_extra_connects]
```

Writes the CPF/IEEE1801 file. The command options help to write the file for CPF as well as for the IEEE1801 standard.

Parameters

`-help`	Prints a brief description that includes the type and default information for each `write_power_intent` parameter. For a detailed description of the command and all its parameters, use the man command: `man write_power_intent`
`file_name`	Specifies name of the IEEE1801 input file. *Data_type*: string, required
For CPF	
`-cpf`	When specified, Innovus writes out the CPF file. *Data_type*: bool, required
`-hinst_only`	Specifies the hierarchical instance name. Writes (partitions) the CPF for the hierarchical instance specified in this option. **Note:** Must be used along with `-cpf` option. *Data_type*: string, optional
`-hinsts`	Writes top-level CPF where the hierarchical instances specified in `-hinsts` option are hard blocks when specified together with `-top_hier -cpf`. *Data_type*: string, optional
`-top_hier`	Writes top-level CPF when specified together with `-hinsts` and `-cpf`. *Data_type*: bool, optional
For IEEE1801	

`-1801`	Writes the IEEE1801 file from the low-power database. It filters out the non-existing instances from the original IEEE1801 file and appends to it the following: • PG connection for the newly-inserted always-on buffers • Nwell/Pwell connections for the low-power cells • `-no_isolation`/`shifter` for the isolation/level shifter strategies for the newly-created ports along the power domain boundary *Data_type*: bool, required
`-incremental_1801`	Specifies the incremental IEEE1801 power intent. *Data_type*: bool, required
`-no_extra_connects`	If this parameter is specified, `write_power_intent -1801` does not write out any extra `connect_supply_net` (PG connection). For example: `write_power_intent -1801 -no_extra_connects out.upf` **Note:** Extra connects means that the extra `connect_supply_net` commands appended by the tool, which are not included in the original UPF. *Data_type*: bool, optional

Related Topics

- `commit_power_intent`
- `delete_power_intent`
- `read_power_intent`
- Low Power Design chapter in the *Innovus User Guide*

write_power_switches

```
  write_power_switches
[-help]
[-power_domain powerDomainName]
[out_file fileName]
```

Exports power switch databases. The need for this command arises in case of defIn-based flows where power switch database required for power switch related operations is not loaded.

So, using `write_power_switches` a user can save power switch database after power switch insertion using `add_power_switches` for later use and it can be loaded back.

Note: The enable net IN and OUT connections should remain the same in the `netlist` in power switch database export/import process and the commands need not be run when `save_design` is run because the power switch database is automatically saved and restored.

PSO File Generation and Usage

A PSO file gets generated as an output of the `write_power_switches` command, which can be used in a future session, if needed. One scenario where this file can be needed is when a DEF file containing a power switch chain is not used in the power switch synthesis and hence the incoming netlist to Innovus would not have any power switches. The power switch chain is then created at the `init` stage of Innovus using the `add_power_switches` command in such a run. A database saved using `write_db` after running the `add_power_switches` command, will have the power switch chain saved in a `<design.pso>` file. When restored, this database can also be used to write out the saved PSO file using the `write_power_switches` command. When the power switches are added using the `add_power_switches` command, the `report_power_switches` command can be used to validate their addition.

To avoid regenerating power switches in future implementation runs, save the power switch chain using the `write_power_switches` command along with the floorplan DEF. Ensure that the `-routing` parameter of the `write_def` command is also used along with `-floorplan` and `-no_std_cells` to save the power switch network under the NETS section of the DEF.

Refer to the following flows to understand the usage of the PSO file:

Innovus Session 1:

```
<init Design1 - incoming design has no power switches>
add_power_switches
write_power_switches test.pso
write_def -floorplan -no_std_cells -routing test.def
```

Innovus Session 2:

```
<init Design2 - incoming design has no power switches>
read_power_switches -in_file test.pso
read_def test.def
```

While reading the DEF file, you can identify the power switches using the database commands, even without using the `read_power_switches` command. Following is a usage example of such a database command:

```
get_db [get_db insts -if {.name == inst1/psoI_PD1_1_HSWX1_200_1582_0} ] .base_cell.is_power_switch
```

Here, `inst1/psoI_PD1_1_HSWX1_200_1582_0` is the instance name of the switch cell.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `write_power_switches` parameter. For a detailed description of the command and all its parameters, use the `man` command: `man write_power_switches`
`-power_domain` *`powerDomainName`*	Specifies the power domain whose power switch information is to be exported. *Data_type*: string, optional
`out_file fileName`	Specifies the name of the export file. *Data_type*: string, optional

Related Topics

- `add_power_switches`
- `check_power_switches`
- `delete_power_switches`
- `move_power_switches`
- `read_power_switches`
- `report_power_switches`
- `update_power_switches`
- Low Power Design chapter in the *Innovus User Guide*

write_power_switches_for_floorplan

```
write_power_switches_for_floorplan
[-help]
out_file
```

Stores the power switch database information, which can later be restored in the same floorplan through the `read_power_switches` command.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `write_power_switches_for_floorplan` parameter. For a detailed description of the command and all its parameters, use the `man` command: `man write_power_switches_for_floorplan`
`out_file`	Specifies the output file for storing the power switch database information. *Data_type*: string, optional

Related Information

- `add_power_switches`
- `read_power_switches`

19

Machine Learning Commands

- init_ml
- write_ml_design
- write_ml_library
- write_ml_timing_graph
- write_ml_timing_path

init_ml

```
init_ml
[-help]
[-comment string]
[-flow_phase string]
[-name string]
-prefix string
[-scenario string]
```

Initializes the environment to start writing data for JedAI. Run this command before using any of the following commands to ensure that content is properly labeled for big data ingestion and registration:

- `write_ml_design`
- `write_ml_library`
- `write_ml_timing_graph`
- `write_ml_timing_path`

Parameters

`-help`	Prints a brief description that includes the type and default information for each `init_ml` parameter. For a detailed description of the command and all of its parameters, use the man command `man init_ml`.
`-comment` *`string`*	Specifies a comment. *Data_type*: string, optional
`-flow_phase` *`string`*	Specifies the flow phase. *Data_type*: string, optional
`-name` *`string`*	Specifies the design name. *Data_type*: string, optional
`-prefix` *`string`*	Specifies the prefix for the output files. *Data_type*: string, required
`-scenario` *`string`*	Specifies the scenario. *Data_type*: string, optional

Example

- The following command the environment to start writing data for JedAI for the `cts` flow phase with the `default` scenario for the `dtmf` design.

  ```
  init_ml -prefix out/ -name dtmf -flow_phase cts -scenario default -comment "This is a demo."
  ```

Related Information

- write_ml_design
- write_ml_library
- write_ml_timing_graph
- write_ml_timing_path

write_ml_design

```
write_ml_design
[-help]
[-no_power]
[-objects object_list]
-prefix string
[-verbose]
[-views view_list]
```

Writes design-related data for JedAI to csv files with specified prefix. It includes data that is design specific and relates to logical, physical, and timing type objects.

Parameters

`-help`	Prints a brief description that includes the type and default information for each `write_ml_design` parameter. For a detailed description of the command and all of its parameters, use the man command `man write_ml_design`.
`-no_power`	Specifies that power attributes should not be dumped. *Data_type*: bool, optional
`-objects` *object_list*	

	Specifies the list of db object(s) to be dumped, which could include one or more of the following values:
	`analysis_view arc base_cell base_pin blockage clock congestion constraint_mode`
	`delay_corner design gcell_grid hinst hnet hpin hport inst layer marker net netlist`
	`opcond option p2p partition patch_wire pg_base_pin pg_pin pin pin_shape port`
	`rc_corner route_rule row site skew_group special_net timing_condition track_pattern unit_parasitic via_def`
	Use curly braces to enclose multiple values.
	Data_type: tcl_obj, optional

`-prefix` *string*	Specifies the prefix for the output files. *Data_type*: string, required
`-verbose`	Displays a more detailed message. *Data_type*: bool, optional
`-views` *view_list*	
	Dumps the data for the specified analysis view(s). *Data_type*: tcl_obj, optional

Example

- The following command writes the attributes of all design objects to csv files with the specified prefix:

```
write_ml_design -prefix out/
```

Related Information

- `init_ml`
- `write_ml_library`
- `write_ml_timing_graph`
- `write_ml_timing_path`

write_ml_library

```
write_ml_library
[-help]
[-objects object_list]
-prefix string
```

Writes library-related data for JedAI to csv files with the specified prefix. It includes data that is library specific and relates to lib_cells, timing_tables, and power_tables.

Parameters

`-help`	Prints a brief description that includes the type and default information for each `write_ml_library` parameter. For a detailed description of the command and all of its parameters, use the man command `man write_ml_library`.
`-objects object_list`	
	Specifies the list of db object(s) to be dumped, which could include one or more of the following values: `lib_arc lib_cell lib_feature lib_pin library library_set power_table timing_table` Use curly braces to enclose multiple values. *Data_type*: tcl_obj, optional
`-prefix string`	Specifies the prefix for the output files. *Data_type*: string, required

Example

- The following command writes library-related data for JedAI to csv files with the specified prefix:

  ```
  write_ml_library -prefix out/
  ```

Related Information

- init_ml
- write_ml_design
- write_ml_timing_graph
- write_ml_timing_path

write_ml_timing_graph

```
write_ml_timing_graph
[-help]
[-noedge]
-prefix string
[-register_only]
[-views view_list]
```

Writes timing-related data for JedAI to csv files. It includes data representing the complete timing graph for a given set of analysis views.

Parameters

`-help`	Prints a brief description that includes the type and default information for each `write_ml_timing_graph` parameter. For a detailed description of the command and all of its parameters, use the man command `man write_ml_timing_graph`.
`-noedge`	Specifies that edge data should not be dumped. This option can be used to save time. *Data_type*: int, optional
`-prefix string`	Specifies the prefix for the output files. *Data_type*: string, required
`-register_only`	Specifies that only nodes for register should be dumped. *Data_type*: bool, optional
`-views view_list`	Dumps the data for the specified analysis view(s). *Data_type*: tcl_obj, optional

Example

- The following command writes the timing-related data for JedAI to csv files with the specified prefix:

```
write_ml_timing_graph -prefix out/
```

Related Information

- `init_ml`
- `write_ml_design`
- `write_ml_library`
- `write_ml_timing_path`

write_ml_timing_path

```
write_ml_timing_path
[-help]
[-anchor_file string]
[-each_endpoint]
[-gen_anchor]
[-hpin]
[-io_only]
[-max_paths int]
[-max_slack float]
[-nworst int]
[-path_group string]
[-per_partition]
-prefix string
[-retime {aocv ssta path_slew_propagation aocv_path_slew_propagation}]
[-start_pathid int]
[-verbose]
[-views view_list]
```

Writes timing-related data for JedAI to csv files. It includes data for individual timing paths. Timing path data can be computed in either graph-based analysis (GBA) or path-based analysis (PBA) fashion using the `-retime` option.

Parameters

`-help`	Prints a brief description that includes the type and default information for each `write_ml_timing_path` parameter. For a detailed description of the command and all of its parameters, use the man command `man write_ml_timing_path`.
`-anchor_file` *string*	Specifies the name of the anchor file to be used as input. *Data_type*: string, optional
`-each_endpoint`	Specifies whether to dump at least one path for each end point. *Data_type*: bool, optional

`-gen_anchor`	Generates the anchor file. *Data_type*: bool, optional
`-hpin`	Dumps hpin in timing points. *Data_type*: bool, optional
`-io_only`	Specifies that only the io path should be dumped. *Data_type*: bool, optional
`-max_paths` *int*	Specifies the max_paths for `report_timing`. *Data_type*: int, optional
`-max_slack` *float*	Specifies the max_slack for `report_timing`. *Data_type*: float, optional
`-nworst` *int*	Specifies the nworst for `report_timing`. *Default*: `100` *Data_type*: int, optional
`-path_group` *string*	Specifies the path_group for `report_timing`. *Data_type*: string, optional
`-per_partition`	Specifies that only paths through top level partitions should be dumped. *Data_type*: bool, optional
`-prefix` *string*	Specifies the prefix for the output files. *Data_type*: string, required
`-retime {aocv ssta path_slew_propagation aocv_path_slew_propagation}`	
	Specifies the method to be used for reanalysis of specified set of paths with `report_timing`. *Data_type*: enum, optional
`-start_pathid` *int*	Specifies the starting path id. *Default*: `1` *Data_type*: int, optional

`-verbose`	Displays a more detailed message. *Data_type*: bool, optional
`-views` *view_list*	Dumps the timing path for the specified analysis view(s). *Data_type*: tcl_obj, optional

Example

- The following command writes out the attributes of timing paths to csv files with the specified prefix:

```
write_ml_timing_path -prefix out/ -max_paths 10000 -nworst 1
```

Related Information

- `init_ml`
- `report_timing`
- `write_ml_design`
- `write_ml_library`
- `write_ml_timing_graph`

20

Metal and Via Fill Commands

- add_metal_fill
- add_metal_fill_signoff
- add_pg_fill
- add_track_fill
- add_via_fill
- delete_metal_fill
- fix_open_fill
- get_metal_fill_spacing_table
- get_pg_fill_config
- get_track_fill_config
- report_metal_fill
- set_metal_fill
- set_metal_fill_spacing_table
- set_pg_fill_config
- set_track_fill_config
- set_via_fill
- trim_metal_fill
- trim_metal_fill_near_net

add_metal_fill

```
add_metal_fill
[-help]
[-area x1 y1 x2 y2 ]
[-extra_critical_nets net_name_list ]
[-exclude_vias via_list ]
[-ignore_lef_density]
[-iteration_names { list_of_iteration_names } ]
[-keep_square_floating]
[-layers layerName | { list_of_layer_names } ]
[-on_cell [-std]]
[-pg_fill]
[-remove_floating_fill]
[-slack_threshold slack ]
[-snap]
[-square_shape] | [-nets { net_name_list } [-generated_via_only] [-mesh | -mesh_only]]
[-stagger {on | off | diag}]
[-timing_aware {on | off | sta}]
[-use_non_default_spacing]
```

Inserts inactive metal into a placed and routed design to achieve the metal density within the range required by a specific manufacturing process. Avoids inserting metal on top of macros if it can achieve the preferred metal density without doing so. Use this command to insert metal fill after timing analysis and post route is complete.

`add_metal_fill` provides complete support for Through Silicon Via (TSV) designs. You can insert tied-off metal fill in back side metal layers and add metal fill in iterations to back side layers.

During timing-driven metal fill, the software assigns costs to signal and clock nets so it can avoid placing metal fill near timing-critical nets.

- It is recommended that you add metal fill after adding via fill. Alternatively, the via fill can also be added between floating metal fill.
- If you use the `-timing_aware sta` parameter, you must run timing analysis before adding metal fill.

This command uses the values specified by the `set_metal_fill` command.

Some technologies use triangular shapes for Corner Stress Relief patterns. However, metal fill does not support triangular route blockages currently.

Adding Metal Fill in Multi-Threading Mode

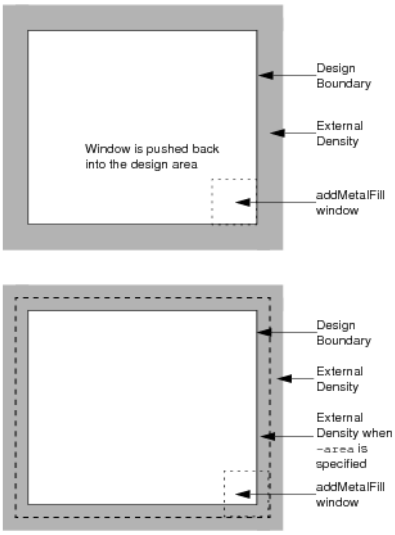
You can add metal fill in multi-threading mode by using the `set_multi_cpu_usage` command before adding the metal fill.

Metal Fill supports the new `set_multi_cpu_usage` command use model. For more information, see the following documents:

- Multiple-CPU Processing Commands chapter of the *Innovus Text Command Reference*.
- Accelerating the Design Process By Using Multiple-CPU Processing chapter of the *Innovus User Guide*.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `add_metal_fill` parameter. For a detailed description of the command and all its parameters, use the `man` command: `man add_metal_fill`

`-area` *x1 y1 x2 y2*	Specifies a set of coordinates within which to add metal fill. This command reduces the fill area by half of the maximum design rule spacing defined in the LEF file. For example, if you specify `-area 0 0 100 100`, and the maximum spacing rule in the LEF file is 2.5 µ, `add_metal_fill` adds fill in the following area: `1.25 1.25 98.75 98.75`. *Default:* If you do not specify this parameter, adds metal fill to the entire design area (taking the maximum spacing rule into account). ***Behavior of add_metal_fill -area with external metal density*** When metal fill is added to a block design the external metal density provides a density assumption to `add_metal_fill`. The external density is used only for density calculation purposes and no fill shapes are added outside of the design area. When the `-area` parameter is not specified, the external density value is not used. In this case, the `add_metal_fill` window is pushed back if it extends outside the design area, as shown in the figure below.  If the `-area` parameter is specified and is larger than the design size then `add_metal_fill` windows extend outside the block area and use the external density value (see `set_metal_fill` for external density value specification) for that area. If external density is less than the specified preferred density, it is more likely that shapes will be added to the area. If external density is greater than the preferred density it is less likely that shapes will be added (shapes are only added within the design boundary). **Note:** Windows outside the design boundary are ignored. *Data_type*: rect, optional

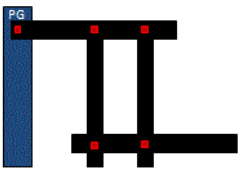
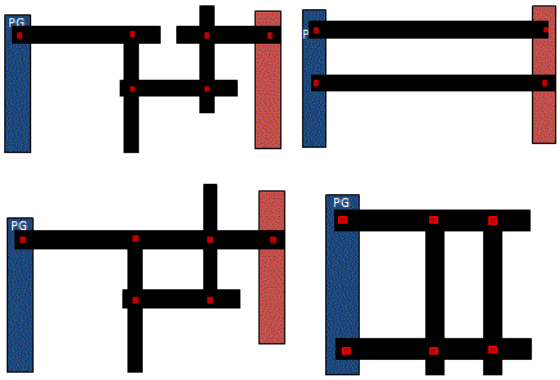
`-exclude_vias` *via_list*	Specifies vias or via rules to exclude from use by metal fill. Use this parameter to prevent the software from using the same vias inside MACROs and for metal fill insertion.
	
	Separate the arguments with spaces. You can specify both via names and via rules.
	• The following example shows how to specify a list of via names:
	`-exclude_vias {VIA12 VIA23}`
	• The following example shows how to specify a list of via rules:
	`-exclude_vias {via1Array via2Array}`
	• The following example shows how to specify both via names and via rules:
	`-exclude_vias { VIA12 VIA23 via1Array via2Array}`
	
	Data_type: string, optional

`-extra_critical_nets` *net_name_list*	Specifies a list of extra critical nets for adding metal fill in the timing-aware mode. If you are using a third-party tool for sign-off static timing analysis (STA), your list of timing critical nets may be different from that generated by `report_timing` in Innovus/Tempus. In such a case, you can use `-extra_critical_nets` in one of the following ways: • In the `-timing_aware sta` mode: The tool adds the extra nets you specify to the Innovus list of critical nets determined via the `-slack_threshold` method. If the list you specify is exactly the same as the Innovus list of critical nets, the tool works in the same way as it would without `-extra_critical_nets`. • In the `-timing_aware on` mode: In this mode, the tool will consider the specified critical net list as medium cost. Use `-extra_critical_nets` in this mode if you want to avoid the runtime hit associated with performing timing analysis using the `-timing_aware sta` option. **Note:** If you use the `-extra_critical_nets` option in the `-timing_aware off` mode, the tool waives the option and issues a warning message. *Data_type*: string, optional
`-generated_via_only`	Specifies that only generated vias are used for metal fill connections. **Note:** This parameter is enabled only when the `-nets` parameter is specified. *Data_type*: bool, optional
`-ignore_lef_density`	Ignores the MACRO `DENSITY` table in the LEF file and instead uses the default macro density calculation method. For more information on the default macro density calculation method, refer to the "*Estimating Density of Blockage*" section in the Optimizing Metal Density chapter of the *Innovus User Guide*. **Note:** If you do not specify this parameter, `add_metal_fill` uses the LEF MACRO density table for density calculations. *Data_type*: bool, optional

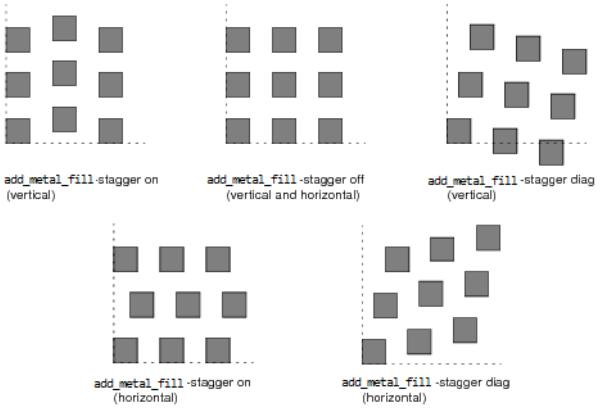
`-iteration_names` {*list_of_iteration_names*}	Specifies a list of iterations (the names are specified by the `set_metal_fill -iteration_names` parameter) for adding metal fill. Processes the iterations in the order listed, and stops when the preferred density is reached in any iteration.
	Running `add_metal_fill` without the `-iteration_names` parameter is the same as running the following command:
	`add_metal_fill -iteration_names default`
	If `add_metal_fill` is looking for `default`, that is, if `-iteration_names default` is specified or if you do not specify the `-iteration_names` parameter, and the default settings are not defined by `set_metal_fill`, the `add_metal_fill` command uses it own internal default values, which are not derived from any `set_metal_fill -iteration_names` name.
	When you run `add_metal_fill` in iteration mode, it runs several times with just one invocation of the command, and uses the ending density from an iteration as the starting density of the next iteration. For that one invocation, the window size and step for a specific layer must be the same for all iterations.
	Because `add_metal_fill` uses preferred density to estimate the macro density, a different preferred density setting in `iteration_name_list will` cause the different density estimation. The same preferred density setting in `iteration_name_list` is recommended.
	In the following example, `add_metal_fill` fails because the specified values for `-window_size` and `-window_step` in `step1`, `step2`, and `step3` are different:
	`set_metal_fill -iteration_names step1 -layers ... \`
	`  -window_size 100 100 -window_step 50 50`
	`set_metal_fill -iteration_name step2 -layers ... \`
	`  -window_size 100 100 -window_step 50 50`
	`set_metal_fill -iteration_names step3 -layers ... \`
	`  -window_size 50 50 -window_step 25 25`
	`add_metal_fill -iteration_names {step1 step2 step3} ...`
	If you run `add_metal_fill` two times, and separate `step3`, (because its values are different from those of `step1` and `step2`) the commands will succeed:
	`add_metal_fill -iteration_names {step1 step2} ...`
	`add_metal_fill -iteration_names step3 ...`
	Data_type: string, optional

`-keep_square_floating`	Specifies that unconnected (floating) metal fill must be square. *Default:* If you do not specify this parameter, the software can create rectangular and square metal fill that is unconnected. *Data_type*: bool, optional

`-layers layerName | {list_of_layer_names}`	Specifies the layers, including back side metal layers, on which the metal fill shapes will be created. If you specify more than one layer, separate each layer with a space and enclose the list in braces or double quotation marks. For example, to specify metal layers 1, 2, and 3, type one of the following commands:
	`add_metal_fill -layers "METAL1 METAL2 METAL3"`
	`add_metal_fill -layers {METAL1 METAL2 METAL3}`
	Default: If you do not specify the `-layers` parameter, Innovus adds metal fill to all layers.
	Data_type: string, optional

`-mesh`	Connects metal fill shapes to the power mesh, allowing it to carry current as part of the power and ground structure. With mesh connections the software uses the maximum number of cuts in vias, based on the intersection area between layers, instead of the following the minimum-cut rule used with tree connections. Increasing the number of cuts helps to reduce IR drop. **Note:** This parameter is enabled only when the `-nets` parameter is specified. The default values for the metal fill commands are optimized for timing, not power. For recommendations for values that are optimized for power, see "Recommendations for Power Strapping Mode" in the Optimizing Metal Density chapter of the *Innovus User Guide*. *Data_type*: bool, optional
`-mesh_only`	Adds only multi-connection PG fill in the design. To improve IR drop, you may want to use tied-off metal fill to connect to power/ground (PG) nets. However, single-connection PG fill does not help in controlling IR drop. `add_metal_fill -mesh_only` enables you to add only mesh or multi-connection fill and removes any single-connection PG fill. As a result, you can improve IR drop and reduce timing impact in the design. With `-mesh_only`, the following PG fills are removed:  The following PG fills are retained with `-mesh_only`:  *Data_type*: bool, optional
`-nets {net_name_list}`	Specifies a list of special nets to which to connect metal fill. The nets must be listed in the `SPECIALNETS` section of the DEF file. PG nets that overlap with the fillwire are sorted by width and length. `add_metal_fill` gives higher priority to the PG nets with greater dimensions (wider width and longer length), irrespective of their sequence in the `net_name_list`. *Default:* If you do not specify this parameter, metal fill is not connected to special nets (it is floating). To specify nets VDD and VSS for connection, type the following command: `add_metal_fill -nets {VDD VSS}` **Note:** Do not specify this parameter if you also specify `-square_shape` as `-nets` only applies rectangular shapes. *Data_type*: string, optional

`-on_cell [-std]`	Allows the software to add metal fill to `CLASS BLOCK` and `CLASS PAD` macro cells. Before adding the metal fill, make sure that the cells are at the preferred density, because adding metal fill shapes inside the cells might lead to timing problems inside the cells. If there are PIN/OBS in a cell, metal fill should be added on cells in layers with `-on_cell` option. The density of macro will be estimated according to OBS/PIN area. If there are no PIN/OBS in a cell, metal fill should be added on cells in layers without `-on_cell` option. **Note:** The software can always add metal fill to `CLASS CORE SPACER` cells, but can add it to `CLASS BLOCK` and `CLASS PAD` cells only when the `-std` parameter is specified. Specify the `-std` parameter as well if you want to add metal fill only to standard cells. *Data_type*: bool, optional
`-pg_fill`	Enables the use of power/ground fill flow. *Data_type*: bool, optional
`-remove_floating_fill`	Removes unconnected metal fill from the design. When the software adds metal fill, it creates both connected and unconnected fill. When you specify `-remove_floating_fill`, it removes all the unconnected fill and leaves only metal fill that is connected. *Default:* If you do not specify this parameter, the software leaves floating metal fill in the design. *Data_type*: bool, optional
`-slack_threshold` *slack*	Specifies a slack threshold, in nanoseconds, for critical nets during timing-driven metal fill when you specify `-timing_aware sta`. All nets with slack less than the `-slack_threshold` are considered critical. For most designs, Cadence recommends you specify a small positive value for the slack threshold, such as 100 picoseconds, which gives you a small safety margin, as in the following command: `add_metal_fill -slack_threshold 0.1` *Default:* `0.0` *Data_type*: float, optional
`-snap`	Snaps edges of metal fill shapes and centers of cuts to a user- defined snap grid that is a multiple of the manufacturing grid. When the shapes are snapped to this grid, the edges of the shapes lie on the manufacturing grid. If the shapes are enclosed by on-Mgrid shapes, the shapes will ignore manufacturing grid check. Other Innovus features share the grid, including verify, power planning and routing, and wire editing. Define the grid using the `set_snap_grid` command. If `set_snap_grid` is not specified, the software snaps the shapes to the manufacturing grid. For more information, see `set_snap_grid`. **Note:** When you specify this parameter, the software assumes that vias from the LEF file meet the snap grid requirements. *Data_type*: bool, optional
`-square_shape`	Specifies that only square-shaped metal fill is added. *Default:* If you do not specify this parameter, both rectangular and square metal fill can be added. **Note:** Do not specify this parameter if you use the `-nets` parameter as `-nets` applies only to rectangular shapes. *Data_type*: bool, optional

`-stagger {on | off | diag}`	Specifies the pattern for adding metal fill.
	Default: `on`
	Specify one of the following parameters:
	• `on`: Adds metal fill shapes in a staggered pattern in the preferred routing direction and in an aligned pattern in the non-preferred direction.
	• `off`: Adds metal fill shapes that line up in both the preferred and non-preferred routing directions, so that metal fill is not staggered.
	• `diag`: Adds metal fill shapes in a staggered pattern in both the preferred and non-preferred directions. Specify the offset for staggering with the following command:
	`set_metal_fill -diag_offset x y`
	For example:
	`set_metal_fill -diag_offset 0.05 0.05`
	add_metal_fill -stagger parameters
	 
	
	Data_type: enum, optional

`-std`	Adds metal fill to standard cells only. **Note:** The `-std` parameter can be used only when the `-on_cell` parameter is used. *Data_type*: bool, optional

`-timing_aware {on | off | sta}`	Specifies whether the software considers the timing impact of adding metal fill.
	Default: `on`
	
	Specify one of the following parameters:
	• `off`: Does not consider timing.
	• `on`: Assigns timing costs to nets and adds metal fill to nets with the lowest timing cost first. Adds less metal fill near clock and signal routing.
	Divides nets into categories based on timing cost. The categories are:
	  ◦ Clock nets (highest cost)
	  ◦ Signal nets (moderate cost)
	  ◦ Power and ground nets--netss marked + USE POWER or + USE GROUND in the DEF file (0 cost)
	• sta: Runs in timing-aware mode using the Innovus common timing engine (CTE) for static timing analysis (STA). To benefit from using the sta parameter, you must have run timing analysis with the report_timing command prior to adding metal fill. If there are no timing structures in memory, the sta parameter behaves like the on parameter.
	
	The `sta` parameter also divides nets into categories, but it uses the timing analysis results for assigning timing costs to nets. It adjusts the costs as a function of the slack (the worst slack has the highest cost) and uses the slackThreshold to separate critical and noncritical signal nets. Critical nets are signal nets with slack less than the threshold, and therefore have a higher cost than noncritical nets.
	
	The categories are:
	  ◦ Clock nets (highest cost)
	  ◦ Critical signal nets (moderate cost)
	  ◦ Non-critical signal nets (small cost)
	  ◦ Power and ground nets--marked `+ USE POWER` or `+ USE GROUND` in the DEF file (0 cost)
	
	ⓘ Specify a value for `-slack_threshold` if you specify `sta`.
	
	Data_type: enum, optional

`-use_non_default_spacing`	Uses nondefault spacing for adding metal fill. Some critical nets use a nondefault rule (NDR) for routing as they may need to have wider wires or wider spacing than other nets. Use the `-use_non_default_spacing` option to extend NDR control to `add_metal_fill`. *Data_type*: bool, optional

Example

- The following command adds connected metal fill to the specified area on *METAL1*, *METAL2*, and *METAL3*, using the values set by the `set_metal_fill` command:

  ```
  add_metal_fill -layers {METAL1 METAL2 METAL3} -area 100 200 300 400 -netss {VDD GND}
  ```

  Here's a sample output from the software:

  ```
  add_metal_fill -q .metalfill_27257.conf
  ************************
  Timing Aware on
  P/G Nets: 321
  Signal Nets: 8784
  Clock Nets: 0
  ************************
  Density calculation ...... Slot : 1 of 1
  Density calculation ...... Slot : 1 of 1
  End of Density Calculation : cpu: 0:00:00, real: 0:00:01, peak: 566.00 megs
  process data during iteration 1 in region 1 of 4.
  process data during iteration 1 in region 2 of 4.
  process data during iteration 1 in region 3 of 4.
  process data during iteration 1 in region 4 of 4.
  process data during iteration 2 in region 1 of 4.
  process data during iteration 2 in region 2 of 4.
  process data during iteration 2 in region 3 of 4.
  process data during iteration 2 in region 4 of 4.
  End metal filling: cpu: 0:00:01.2, real: 0:00:02.0, mem 361.10 megs.
  ```

- The following set of commands:
  - Creates a nondefault rule `testndr` that sets the spacing of all layers to `0.8` micrometers.
  - Sets the nondefault rule `testndr` for the `clk` net.
  - Specifies that the router should strictly honor nondefault spacing.
  - Adds metal fill using nondefault spacing
  - Checks the nondefault rule spacing values for every route of the net `clk`.

  ```
  initNdr

  setNdrSpacing 0.8

  createNdr testndr

  setAttribute -nets clk -non_default_rule testndr

  set_db route_strict_honor_non_default_rule 1

  add_metal_fill -use_non_default_spacing

  verifyGeometry -use_non_default_spacing
  ```

- The following command inserts PG fill in back side layers `B1` and `B2`:

  ```
  add_metal_fill -nets {VDD VSS} -layers {B1 B2}
  ```

- The following command saves run time by adding metal fill in iterations to back side layers `B1` and `B2`.

  ```
  add_metal_fill -iteration_names {iter1 iter2} -layers {B1 B2}
  ```

- The following set of commands inserts metal fill in three iterations`-step1`, `step2`, and `stepopc`-to layers metal`1` to metal`8` in timing-aware mode:

  ```
  read_db

  time_design -post_route -report_dir prefill
  ```

```
set_metal_fill -layers {1 2 3 4 5 6 7 8} -iteration_name step1 /
             -gap_spacing 0.140 -min_width 1 -decrement 0.050 /
             -min_length 2 -max_width 5 -max_length 20 -active_spacing 0.8 /
             -window_size 50 50 -window_step 25 25 -min_density 15 /
             -max_density 80 -preferred_density 35 -diagonal_offset 1 1

set_metal_fill -layers {1 2 3 4 5 6 7 8} -iteration_name step2 /
             -gap_spacing 0.140 -min_width 0.140 -decrement 0.010 /
             -min_length 0.14 -max_width 1 -max_length 10 -active_spacing 0.2 /
             -window_size 50 50 -window_step 25 25 -min_density 15 /
             -max_density 80 -preferred_density 25 -diagonal_offset 0.2 0.2

set_metal_fill -layers {1 2 3 4 5 6 7 8} -iteration_name stepopc -opc /
             -gap_spacing 0.07 -min_width 0.07 -decrement 0.010 /
             -min_length 0.07 -max_width 1 -max_length 10 -active_spacing 0.07 /
             -opcactive_spacing 0.1 -window_size 50 50 -window_step 25 25 /
             -min_density 15 -max_density 80 -preferred_density 25
add_metal_fill -layers {1 2 3 4 5 6 7 8} /
             -iteration_names {step1 step2 stepopc} /
             -timing_aware sta
```

Related Information

Optimizing Metal Density chapter of the *Innovus User Guide*

add_metal_fill_signoff

```
add_metal_fill_signoff
[-help]
[-and_area_size size_value]
[-area {x1 y1 x2 y2}]
[-attach_instance_name attr_num]
[-attach_net_name attr_num]
[-attach_net_prop {{prop_name attr_num} ... }]
[-auto_load_fills]
[-background_mode]
[-clock]
[-common_blockage {layer_num1 layer_datatype1 ...}]
[-control file_name]
[-delete_area {{layer_list1}{x1 y1 x2 y2}...}]
[-delete_point {{layer_list1}{x1 y1}...}]
[-die_area_as_boundary]
[-distributed n]
[-distributed_timeout value]
[-excluded_area_size size_value]
[-fill | -incremental | -trim | -flatten | -delete_fill | -merge_fill_edits | -view_fills | -delete_flatten_fills]
[-fill_cell cell_name]
[-fill_layer {layers_list}]
[-fill_output_mode {gdsii | oasis}]
[-layer_map_file streamOut_map_file]
[-license_timeout]
[-license_distributed_continue]
[-lib_name library_name]
[-lsf]
[-master_lsf]
[-merge list_of_external_stream_files]
[-min_density percent1 layer_list1]
[-net net_names]
[-no_structure_name]
[-no_via_fills]
[-offset {x y}]
[-output_macros]
[-report_file]
[-rule_file rule_file_name]
[-skip_map_check]
[-slack_threshold float]
[-spacing float]
[-spacing_above float]
[-spacing_below float]
[-stripes number]
[-structure_name structureName]
[-tech_lib string]
[-tech_set string]
[-tech string]
[-tmp_working_dir work_dir]
[-trim_effort {high | med | low}]
[-trim_layer {layers_list}]
[-trim_metal_fill spacing_file]
[-union_density percent1 layer_list1]
[-uniquify_cell_names]
[-units {100 | 200 | 400 | 800 | 1000 | 2000 | 4000 | 8000 | 10000 | 20000}]
[-window_size float]
[-window_step float]
```

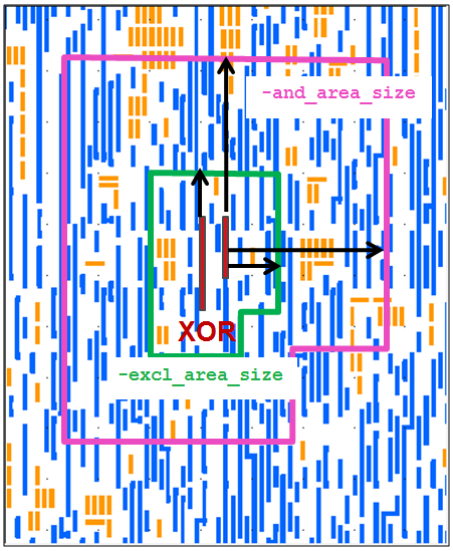
Runs the signoff metal fill flow as per specified primary and sub options.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `add_metal_fill_signoff` parameter. For a detailed description of the command and all its parameters, use the `man` command: `man add_metal_fill_signoff`
`-and_area_size` *`size_value`*	Specifies a value for sizing up the changed area to use shapes within this area. This parameter helps in calculating the density and creating areas to be filled, if they fall below the minimum density. The default value of this parameter is 10. A value lesser than the default value may be used to improve the flow performance, if no accuracy issues occur. A value greater than the default value can be used to fix the accuracy issues. The minimum value of this parameter is `0.001` and maximum value is `1000`.
`-area {x1 y1 x2 y2}`	Specifies a set of coordinates within metal fill can be trimmed.
`-attach_instance_name` *`attr_num`*	Specifies the instance name to attach.
`-attach_net_name` *`attr_num`*	Specifies the net name to attach.
`-attach_net_prop {{`*`prop_name`* `attr_num} ...}`	
	Specifies the net property to attach.
`-auto_load_fills`	Updates the FILL data automatically in the software after when the `add_metal_fill_signoff` command modifies the HMF database. In case the HMF database is not modified, the FILL data will be hidden in the software's design.
`-background_mode`	Runs metal fill in the background mode without locking up the software license. **Note:** All the metal fill options running in the background must be used with the `-background_mode` parameter.
`-clock`	Specifies whether to trim metal fill around all clock nets.

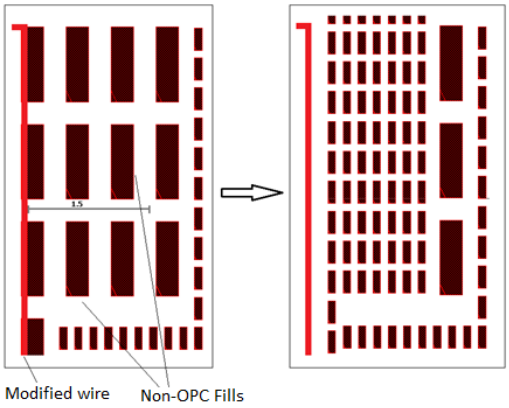
`-common_blockage {`*`layer_num1 layer_datatype1`* `...}`	Specifies common blockage fill layers. This allows you to specify the blockage layer only once, instead of multiple individual blockages for the metal fill layers. The parameter value format is `{`*`layer_number layer_datatype`*`}`. You can also specify multiple layers, and each blockage layer should specify a set of two values. The *`layer_number`* must be an integer. Following are the examples of the usage of this parameter:
	`add_metal_fill_signoff -common_blockage {150 28}`
	`add_metal_fill_signoff -common_blockage {150 28 120 55}`

`-control` *`file_name`*	Specifies the control file, which is created automatically when a job is submitted from the PVS GUI. This parameter can be used, if you have run a directory submitted from the GUI. Use this parameter to rerun the PVS job from the command line and get the results, which are exactly the same as the GUI submission. The control file is placed in the run directory with the `pvsdrcctl` name and has only the command entries for certain fields in the GUI. You must not modify the file by adding commands or parameters that are not available in the GUI. If you do that, the commands can be ignored by the PVS rule compiler or the results can be indeterminate.
`-delete_area {{`*`layer_list1`*`} {`*`x1 y1 x2 y2`*`} ...}`	Specifies the x and y coordinates of the area where metal fill is deleted. Following is the usage example of this parameter: `add_metal_fill_signoff -delete_area {{M1 M2 M6} {0 0 50 50}}`
`-delete_fill`	Deletes metal fills in the specified window or coordinates.
`-delete_flatten_fills`	Deletes flattened metal fills without updating the HMF database.
`-delete_point {{`*`layer_list1`*`}{`*`x1 y1`*`}...}`	Specifies the x and y coordinates of the point where metal fill is deleted. Following is the usage example of this parameter: `add_metal_fill_signoff -delete_point {{M2} {134.7 14.21}}`
`-die_area_as_boundary`	Specifies the die area as boundary.

`-distributed n`	Specifies the number of local cores to use. The default value is fetched from the EDI settings.
`-distributed_timeout value`	Specifies the time to wait for the slave processes to start.
`-excluded_area_size size_value`	Specifies the area to be created around the metal fill area to check for the existing metal fill shapes and to avoid creation of metal fill shapes too close to the existing metal fill shapes. The default value of this parameter is `1.6`. A value lesser than the default value may be used to improve the flow performance, if no accuracy issues occur. A value greater than the default value can be used to fix the accuracy issues. The minimum value of this parameter is `0.001` and maximum value is `1000`. The following picture shows the usage of `-and_area_size` and `-excluded_area_size` parameters: 
`-fill`	Runs the signoff metal fill flow to generate hierarchical metal fills database.
`-fill_cell cell_name`	Specifies metal fill cell name.
`-fill_layer {layers_list}`	Fills only a set of specified metal and/or via metal layers. Following is the usage example of this parameter: `add_metal_fill_signoff -fill_layer {M1 VIA12 M2}`
`-fill_output_mode {gdsii | oasis}`	Generates metal fill database in the `gdsii` or `oasis` format. The default format is `gdsii`.
`-flatten`	Flattens metal fills database and loads into the software.
`-incremental`	Runs incremental metal fill post ECO.
`-layer_map_file streamOut_map_file`	Specifies the `streamOut` metal layer map file.
`-license_timeout`	Specifies the time (in seconds) to wait if the software license is not immediately available while initiating the run. The default value of this parameter is 300 seconds.
`-license_distributed_continue`	Initiates the software run using fewer available CPUs based on the number of licenses obtained. Use this parameter in case PVS cannot obtain the required number of licenses in the multiprocessing mode.
`-lib_name library_name`	Specifies the library name.

`-lsf`	Distributes the processing of the software run to different CPUs using LSF. While using this parameter, the `DP_LSF_RESOURCE` environment variable must be set to specify the LSF resource string, as shown below: `setenv DP_LSF_RESOURCE 'resource string for slave processes'` Additionally, the `-tmp_working_dir` *`work_dir`* should be used to specify local working directory for the LSF run.
`-master_lsf`	Submits the master PVS process on LSF. While using this parameter, the `MASTER_LSF_RESOURCE` environment variable should be set to specify LSF resource string, as shown below: `setenv MASTER_LSF_RESOURCE 'resource string for master process'`
`-merge` *`list_of_external_stream_files`*	
	Merges the external stream files.
`-merge_fill_edits`	Updates metal fill database with the metal fill edits executed in the software.
`-min_density percent1` *`layer_list1`*	Specifies percentage value of the minimum metal density. The default value of this parameter is `0` for each metal layer. This parameter trims metal fill below when the specified window density percentage could cause any metal density violations. This parameter is controlled by the `-window_size` and `-window_step` values, as shown in the below example: `add_metal_fill_signoff -min_density {20 {M1 M2} 35 M3}`
`-net net_names`	Specifies a list of critical net names or pointers, which are surrounded by the metal fill area.
`-no_structure_name`	Merges the GDSII file without loading the design.
`-no_via_fills`	Specifies not to generate metal via fills.
`-offset {`*`x y`*`}`	Defines the offset displacement to be applied.
`-output_macros`	Displays the LEF abstract information about macros, such as LEF pin geometries and obstructions.
`-report_file`	Displays the cell name, which is changed to file.
`-rule_file` *`rule_file_name`*	Specifies metal fill rule file name.
`-skip_map_check`	Ignores the cross-referencing of the `-layer_map_file` parameter and the Innovus layer names.
`-slack_threshold` *`float`*	Trims any critical net(s) in the timing aware flow with the slack lesser than the specified value (in nanoseconds). Following is the usage example of this parameter: `add_metal_fill_signoff -slack_threshold 0.1`
`-spacing` *`float`*	Trims metal fill on the specified metal layer around critical net routes. **Note:** The `-spacing` and `-trim_effort` parameters are mutually exclusive.
`-spacing_above` *`float`*	Trims metal fill on above the specified metal layer around critical net routes.
`-spacing_below` *`float`*	Trims metal fill on below the specified metal layer around critical net routes.
`-stripes` *`number`*	Specifies the number of stripes that divide the design. The default is value of this parameter is `1`.
`-structure_name` *`structureName`*	Specifies the superstructure that consists of all the geometries that the map file writes out.
`-tech_lib` *`string`*	Specifies the PVS technology library for `-fill` and `-incremental`.
`-tech_set` *`string`*	Specifies the PVS technology set for `-fill` and `-incremental`.

`-tech` *`string`*	Specifies the PVS technology to use for identifying the sign-off rules file. **Note:** You can use this parameter with either the `-rule_file` or `-tech` parameter.
`-tmp_working_dir` *`work_dir`*	Specifies the working directory for the background or LSF run.
`-trim`	Trims metal fills in the metal fills database for timing. This parameter is density-related.
`-trim_effort {high | med | low}`	Trims metal fill area with the high, medium, or low effort around the critical net routes. This parameter also helps in determining the blockage size for trimming metal fill. Following are the parameter values: • `high`: Presents area of two times of the layer pitch value. • `med`: : Presents area of 1.5 times of the layer pitch value. • `low`: : Presents the same area of the layer pitch value. Following is the usage example of this parameter: `add_metal_fill_signoff -trim_effort med` Here, if the M2 layer pitch is `0.8 um`, M2 fills will be trimmed around selected nets within the distance of `1.2 um`. **Note:** The `-trim_effort` and `-spacing` parameters are mutually exclusive.
`-trim_layer {`*`layers_list`*`}`	Trims only the set of specified layers. Following is the usage example of this parameter: `add_metal_fill_signoff -trim_layer {M1 M2}`

`-trim_metal_fill` *spacing_file*	Specifies the spacing file containing information related to trim metal fill in the incremental run. Spacing file contains spacing values between the routed metal, non-OPC and OPC metal fill layers. Metal fills are trimmed around ECO changes according to the specified spacing values, post which incremental run generates fill data in the trimmed area. Following is the format of the spacing file:
	`#layer f-a o-a`
	`M1 0.60 0.16`
	`M2 0.60 0.16`
	`M3 0.60 0.16`
	`M4 0.60 0.16`
	`M5 0.60 0.16`
	`M6 0.60 NA`
	Here:
	• `Mx`: Specifies the LEF layer name.
	• `f-a`: Specifies the metal fill active spacing.
	• `o-a`: Specifies the is OPC fill active spacing.
	• `NA`: Refers to no trimming.
	The following picture shows the usage of `-trim_metal_fill` parameter:
	 
	
	The spacing file for the above example has the following settings for the M2 layer:
	`M2 1.5 0.6`
	In this example, the non-OPC metal fills, which are 1.5 um or closer to the modified wire, are trimmed. Additionally, the metal fill data is regenerated in the trimmed area.

`-union_density {`*percent1 layer_list1* `[{`*percent2 layer_list2*`}]...}`	
	Specifies percentage value of the union metal density. The default value of this parameter is `0` for each metal layer. Union density for a layer is the sum of metal density of the current layer, layer below, and layer above the specified windows. Metal fills are trimmed not to violate the `-union_density` settings for the current metal. Following is the usage example of this parameter: `add_metal_fill_signoff -union_density {20 M1}`
`-uniquify_cell_names`	Creates unique cell names, if there are name collisions while using the `-merge` parameter.
`-units {100 | 200 | 400 | 800 | 1000 | 2000 | 4000 | 8000 | 10000 | 20000}`	
	Specifies the resolution of the input/output file. The default value of this parameter is obtained from software database.
`-view_fills`	Loads the existing HMF database into the software to view the metal fills.

`-window_size float`	Specifies the size of the window, which the metal trim uses to examine the metal density. The default value of this parameter is `100`. Following is the usage example of this parameter: `add_metal_fill_signoff -window_size 50`
`-window_step float`	Specifies the stepping distance between two successive positions of the window for a stepped-window trim density check. The default value of this parameter is `50`. Following is the usage example of this parameter: `add_metal_fill_signoff -window_step 25`

Examples

- The following command runs the background LSF job by setting the resource master and slave LSF resource string before launching the software:
  ```
  setenv MASTER_LSF_RESOURCE'-q lnx64 -W "40:00" -n 4 -P "PVS15.23:PE:test"'
  setenv DP_LSF_RESOURCE'-q lnx64 -W "15:00" -P "PVS15.23:PE:test"'
  ```
- The following commands launch a background LSF job in both the master and slave on LSF:
  ```
  Puts "Launching background lsf job; both master and slave on lsf"
  set Bgdir "./LSF_RUN"
  add_metal_fill_signoff -fill -background_mode -master_lsf -lsf -tmp_working_dir $Bgdir
  ```
  It starts HMF generation in the working directory. Post that, you can choose to continue working in the software or exit from the session. The software license is released, as shown below:

  ```
  ################################################################################
  ###### Running add_metal_fill_signoff
  ################################################################################
  ###### Mode '-fill background mode' on cell 'tdsp'
  ################################################################################

  ######  FILL DB Preparing design data
  Ending "FILL DB Preparing design data" (total cpu=0:00:00.2, real=0:00:00.0, peak res=550.0M, current mem=550.0M)

  ######  FILL DB Submitting -fill background run on LSF
  Job <724739> is submitted to queue <lnx64>.
  Ending "FILL DB Submitting -fill background run on LSF" (total cpu=0:00:00.0, real=0:00:01.0, peak res=550.0M, current mem=532.5M)
  Working directory: "/lan/dfm/pvsqa_test01/PVS/HMF_TESTING/FILL_RUN"

  ################################################################################
  Ending "FILL DB add_metal_fill_signoff" (total cpu=0:00:00.3, real=0:00:01.0, peak res=550.0M, current mem=532.5M)
  ```
- The following command checks the status of the background job:
  ```
  add_metal_fill_signoff -check_background_run $Bgdir
  ```
  Following is the output of the above command:

Run is in progress:

```
#################################################################################
###### Running add_metal_fill_signoff
#################################################################################
BG run information:
 fill_cell:       'tdsp'
 fill_layer:      {M1 M2 M3 M4 M5 M6}
 output_format:   'gdsii'
 LSF job queue:   'lnx64'
 LSF job id:      '724739'

Checking status of BG run:
 run status:      IN PROGRESS
 operation group: 194/1815
 output files:    NOT READY
```

Run completed:

```
#################################################################################
###### Running add_metal_fill_signoff
#################################################################################
BG run information:
 fill_cell:       'tdsp'
 fill_layer:      {M1 M2 M3 M4 M5 M6}
 output_format:   'gdsii'
 LSF job queue:   'lnx64'
 LSF job id:      '724739'

Checking status of BG run:
 run status:      COMPLETED
 operation group: 1815/1815
 output files:    READY
#################################################################################
Ending "FILL DB add_metal_fill_signoff" (total cpu=0:00:00.0, real=0:00:01.0, peak res=533.3M, current mem=533.2M)
```

- The following command restores initial software database link and generates HMF:

```
add_metal_fill_signoff -load_fill_db $Bgdir
```

- The following command kills the background run that is in progress:

```
add_metal_fill_signoff -kill_bg_run $Bgdir
```

  Following is the output of the above command:

```
#################################################################################
###### Running add_metal_fill_signoff
#################################################################################
Job <724739> is being terminated
#################################################################################
Ending "FILL DB add_metal_fill_signoff" (total cpu=0:00:00.0, real=0:00:01.0, peak res=533.6M, current mem=533.6M)
```

- The following command presents an LSF example where master is not on LSF:

```
setenv MASTER_LSF_RESOURCE '-q lnx64 -W "40:00" -n 4 -P "PVS:15.23:PE:test"'
setenv DP_LSF_RESOURCE '-q lnx64 -W "15:00" -P "PVS:15.23:PE:test"'
add_metal_fill_signoff -fill -background_mode -lsf -tmp_working_dir LSF_RUN1
```

  Following is the output of the above command:

```
#################################################################
Get environment info ...
#################################################################
ENV VAR: TCL_LIBRARY = /grid/cic/cm_t3nb_005/PVS/pvs_1611_ehflw/inst_hier/lnx
EV/share/lib/tcltk/08.64/tcl8.6
ENV VAR: TCLX_LIBRARY = /grid/cic/cm_t3nb_005/PVS/pvs_1611_ehflw/inst_hier/ln
DEV/share/lib/tclx/08.40
ENV VAR: TCLLIBPATH = /grid/cic/cm_t3nb_005/PVS/pvs_1611_ehflw/inst_hier/lnx8
V/share/lib/itcl/3.4.1 /grid/cic/cm_t3nb_005/PVS/pvs_1611_ehflw/inst_hier/lnx
EV/share/lib/tclxml/3.2 /grid/cic/cm_t3nb_005/PVS/pvs_1611_ehflw/inst_hier/ln
DEV/share/lib/tcllib/1.18 /grid/cic/cm_t3nb_005/PVS/pvs_1611_ehflw/inst_hier/
1/DEV/share/lib/itcl/3.4.1 /grid/cic/cm_t3nb_005/PVS/pvs_1611_ehflw/inst_hier
-1/DEV/share/lib/tclxml/3.2 /grid/cic/cm_t3nb_005/PVS/pvs_1611_ehflw/inst_hie
7-1/DEV/share/lib/tcllib/1.18-
ENV VAR: LSF_ENVDIR = /grid/sfi/farm/CLUSTERS/sjlsf01/7.0.3/conf
ENV VAR: LSF_SERVERDIR = /grid/sfi/farm/CLUSTERS/sjlsf01/7.0.3/7.0/linux2.6-g
ENV VAR: DP_LSF_RESOURCE = -q lnx64 -W "15:00" -P "PVS:15.23:PE:test"

Available Resource: 1 processors
```

- The following command checks the run status in the above example:

```
add_metal_fill_signoff -check_background_run LSF_RUN1
```

  Following is the output of the above command:

```
<CMD> add_metal_fill_signoff -check_bg_run LSF_RUN1

#########################################################
###### Running add_metal_fill_signoff
#########################################################

BG run information:
  fill_cell:          'tdsp'
  fill_layer:         {M1 M2 M3 M4 M5 M6}
  output_format:      'gdsii'
  host:               'sj-amisha1'
  PVS pid:            '353'

Checking status of BG run:
  run status:         IN PROGRESS
  operation group:    DATA PREPROCESSING
  output files:       NOT READY
```

- The following command presents an LSF example where master is on LSF:

  `add_metal_fill_signoff -fill -background_mode -master_lsf -lsf -tmp_working_dir LSF_RUN2`

  Following is the output of the above command:

```
#########################################################
Get environment info ...
#########################################################
ENV VAR: TCL_LIBRARY = /grid/cic/cm_t3nb_005/PVS/pvs_1611_ehflw/inst_
EV/share/lib/tcltk/08.64/tcl8.6
ENV VAR: TCLX_LIBRARY = /grid/cic/cm_t3nb_005/PVS/pvs_1611_ehflw/inst
DEV/share/lib/tclx/08.40
ENV VAR: TCLLIBPATH = /grid/cic/cm_t3nb_005/PVS/pvs_1611_ehflw/inst_h
V/share/lib/itcl/3.4.1 /grid/cic/cm_t3nb_005/PVS/pvs_1611_ehflw/inst_
EV/share/lib/tclxml/3.2 /grid/cic/cm_t3nb_005/PVS/pvs_1611_ehflw/inst
DEV/share/lib/tcllib/1.18 /grid/cic/cm_t3nb_005/PVS/pvs_1611_ehflw/in
1/DEV/share/lib/itcl/3.4.1 /grid/cic/cm_t3nb_005/PVS/pvs_1611_ehflw/i
-1/DEV/share/lib/tclxml/3.2 /grid/cic/cm_t3nb_005/PVS/pvs_1611_ehflw/
7-1/DEV/share/lib/tcllib/1.18-
ENV VAR: LSF_ENVDIR = /grid/sfi/farm/CLUSTERS/sjlsf01/10.1//conf
ENV VAR: LSF_SERVERDIR = /grid/sfi/hpc/lsf/10.1/linux2.6-glibc2.3-x86
ENV VAR: DP_LSF_RESOURCE = -q lnx64 -W "15:00" -P "PVS:15.23:PE:test"
ENV VAR: LSB_JOBID = 800272

Available Resource: 16 processors
```

- The following command checks the run status in the above example:

  `add_metal_fill_signoff -check_background_run LSF_RUN2`

  Following is the output of the above command:

```
<CMD> add_metal_fill_signoff -check_bg_run LSF_RUN2

#########################################################
###### Running add_metal_fill_signoff
#########################################################

BG run information:
  fill_cell:          'tdsp'
  fill_layer:         {M1 M2 M3 M4 M5 M6}
  output_format:      'gdsii'
  LSF job queue:      'lnx64'
  LSF job id:         '800272'

Checking status of BG run:
  run status:         COMPLETED
  operation group:    75/75
  output files:       READY

#########################################################
```

Related Information

- get_metal_fill_signoff_mode
- set_metal_fill_signoff_mode

add_pg_fill

```
add_pg_fill
[-help]
[-attach_instance_name attr_num]
[-attach_net_name attr_num]
[-attach_net_prop {{prop_name attr_num} ... }]
[-bsub resource_string]
[-core_license license_combination]
[-die_area_as_boundary]
[-distributed n]
[-distributed_timeout seconds]
[-extra_options options]
[-fix_post_process_drc]
[-format {stream oasis}]
[-incremental]
[-initial_working_dir string]
[-layer_map_file streamOut_map_file]
[-lib_name library_name]
[-license_distributed_continue]
[-license_timeout seconds]
[-merge list_of_external_stream_files]
[-nc resource_string]
[-no_structure_name]
[-offset {x y}]
[-output_macros]
[-pg_fill_config_file file_name]
[-pg_hookup_flow {ALL | PG_ONLY | FLOAT_ONLY}]
[-qsub resource_string]
[-report_file]
[-stripes number]
[-structure_name structureName]
[-turbo jobs_num]
[-uniquify_cell_names]
[-units {100 | 200 | 400 | 800 | 1000 | 2000 | 4000 | 8000 | 10000 | 20000}]
[-working_dir working_directory]
[-xor_via_layers]
```

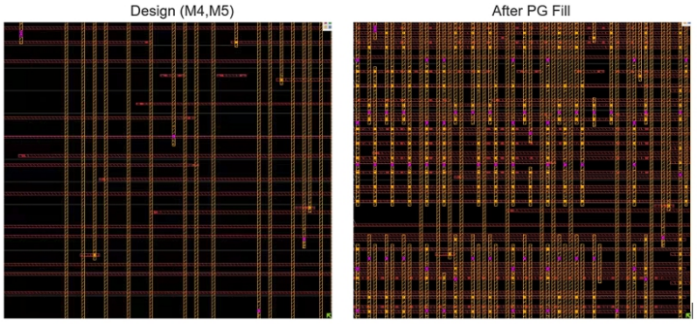
Runs Pegasus PG Fill with default settings and any specified options. This command adds only the floating Track-based Fill (TBF).

Pegasus PG Fill

Pegasus PG Fill is an in-design software used to insert fills connected to Power and/or Ground. PG Fill is generally used for power integrity. These fills are available on the design routing tracks and follow the same DRC rules as design. Pegasus PG Fill:

- Improves IR drop in worst scenarios
- Ensures no negative slack after PG Fill insertion
- Keeps power increase under control

The following example design of the M4 and M5 layers in which fills are inserted post Pegasus PG Fill processing:



Parameters

`-help`	Outputs a brief description that includes the type and default information for each `add_pg_fill` parameter. For a detailed description of the command and all its parameters, use the `man` command: `man add_pg_fill`
`-attach_instance_name` *attr_num*	Specifies the instance name to be attached. *Data_type*: int, optional
`-attach_net_name` *attr_num*	Specifies the net name to be attached. *Data_type*: int, optional
`-attach_net_prop {{`*prop_name attr_num*`} ... }`	Specifies the net property to be attached. *Data_type*: string, optional
`-bsub` *resource_string*	Runs the fill process through BSUB with the specified resource string. *Data_type*: string, optional
`-core_license` *license_combination*	Sets an exact combination of licenses to be used. *Data_type*: string, optional
`-die_area_as_boundary`	Writes the die area as a polygon or a boundary. When this parameter is not specified, the `write_stream` command writes the die area as a rectangle or a box. *Data_type*: bool, optional
`-distributed` *n*	Runs Pegasus in a multi-processor mode. The *n* variable specified with this parameter presents the number of processors. By default, this value is taken from the Innovus multi-CPU usage. *Data_type*: int, optional
`-distributed_timeout` *seconds*	Specifies the wait time for the DP processes to start. *Data_type*: int, optional
`-extra_options` *options*	Specifies the additional run options. *Data_type*: string, optional
`-fix_post_process_drc`	Runs the fill clean-up utility to resolve the DRC issues, which appear post the fill steps. *Data_type*: bool, optional
`-format {stream oasis}`	Specifies the format of the flow. *Data_type*: enum, optional
`-incremental`	Runs Pegasus incremental PG fill. **Note**: This parameter is used with the `-initial_working_dir` and `-working_dir` parameters. *Data_type*: bool, optional
`-initial_working_dir` *string* `-initial_working_dir` *string*	Specifies the working directory with taking information from the previous run. *Data_type*: string, optional
`-layer_map_file` *streamOut_map_file*	Specifies the name of the `write_stream` layer map file. *Data_type*: string, optional
`-lib_name` *library_name*	Specifies the library name. *Data_type*: string, optional
`-license_distributed_continue`	Specifies that if the required number of licenses cannot be obtained in the DP processes, the software continues with fewer CPUs based on the number of licenses obtained. *Data_type*: bool, optional
`-license_timeout` *seconds*	Specifies the wait time (in seconds) to run the software, if a license is not immediately available. *Default*: `300` *Data_type*: int, optional

`-merge` *list_of_external_stream_files*	Specifies the list of external stream files. *Data_type*: string, optional
`-nc` *resource_string*	Runs the fill process through NC with the specified resource string. *Data_type*: string, optional
`-no_structure_name`	Merges the GDSII file without loading the software database. **Note:** This parameter is used with the `-merge` parameter. *Data_type*: bool, optional
`-offset {`*x y*`}`	Specifies the location (in microns) of the design in the GDSII output file. *Data_type*: point, optional
`-output_macros`	Displays the LEF abstract information, such as LEF pin geometries and obstructions for macros. *Data_type*: bool, optional
`-pg_fill_config_file` *file_name*	Specifies the configuration file for the PG fill flow. Any overrides to the default behavior are written in this file. *Data_type*: string, optional
`-pg_hookup_flow {ALL | PG_ONLY | FLOAT_ONLY}`	Specifies the type of fills to be generated. *Default*: `PG_ONLY` *Data_type*: enum, optional
`-qsub` *resource_string*	Runs the fill process through QSUB with the specified resource string. *Data_type*: string, optional
`-report_file`	Specifies the cell name to be changed to the file. *Data_type*: bool, optional
`-stripes` *number*	Specifies the number of stripes that divide the design. *Default*: 1 *Data_type*: int, optional
`-structure_name` *structureName*	Specifies the superstructure that consists of all the geometries that the map file writes out. *Data_type*: string, optional
`-turbo` *jobs_num*	Speeds up the fill process by running the specified number of parallel fill jobs. *Data_type*: int, optional
`-uniquify_cell_names`	Creates unique cell names if there are name collisions, when the software merges files. *Data_type*: bool, optional
`-units {100 | 200 | 400 | 800 | 1000 | 2000 | 4000 | 8000 | 10000 | 20000}`	
	Specifies the resolution for values in the GDSII/OASIS file. *Data_type*: enum, optional
`-working_dir` *working_directory*	Specifies the working directory. *Data_type*: string, optional
`-xor_via_layers`	Switches on comparison of via during incremental fill. **Note:** Via-XOR consumes more time on large databases. *Data_type*: bool, optional

add_track_fill

```
add_track_fill
[-help]
[-attach_instance_name attr_num]
[-attach_net_name attr_num]
[-attach_net_prop {{prop_name attr_num} ... }]
[-bsub resource_string]
[-core_license license_combination]
[-die_area_as_boundary]
[-distributed n]
[-distributed_timeout seconds]
[-extra_options options]
[-fix_post_process_drc]
[-format {stream oasis}]
[-incremental]
[-initial_working_dir string]
[-layer_map_file streamOut_map_file]
[-lib_name library_name]
[-license_distributed_continue]
[-license_timeout seconds]
[-merge list_of_external_stream_files]
[-nc resource_string]
[-no_structure_name]
[-offset {x y}]
[-output_macros]
[-qsub resource_string]
[-report_file]
[-stripes number]
[-structure_name structureName]
[-track_fill_config_file file_name]
[-turbo jobs_num]
[-uniquify_cell_names]
[-units {100 | 200 | 400 | 800 | 1000 | 2000 | 4000 | 8000 | 10000 | 20000}]
[-working_dir working_directory]
```

Adds the Track-Based Fill (TBF) flow to the metal fill layers. Pegasus TBF flow is designed to reuse empty tracks for metal filling. It provides a better fill rate than the signoff fill, because the TBF fills are more homogeneous to the design. The TBF solution is timing aware and DRC-neutral. The flow can be run at any stage of the design and does not require any manual input, since all the required information is fetched from the Innovus database.

The TBF flow also allows altering the default TBF behavior through the various configuration parameters. It supports incremental mode to update TBF fills during the ECO cycles. For better performance and scalability, the TBF flow supports partitioning through the `-turbo` parameter, and job distribution through the `-bsub`, `-qsub`, and `-nc` parameters.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `add_tra` For a detailed description of the command and all its parameters, use the `man` command: `man add_track_fill`
`-attach_instance_name attr_num`	Specifies the instance name to be attached. *Data_type*: int, optional
`-attach_net_name attr_num`	Specifies the net name to be attached. *Data_type*: int, optional
`-attach_net_prop {{prop_name attr_num} ... }`	Specifies the net property to be attached. *Data_type*: string, optional
`-bsub resource_string`	Runs the fill process through BSUB with the specified resource string. *Data_type*: string, optional
`-core_license license_combination`	Sets an exact combination of licenses to be used. *Data_type*: string, optional

`-die_area_as_boundary`	Writes the die area as a polygon or a boundary. When this parameter is not specified, the *Data_type*: bool, optional
`-distributed n`	Runs Pegasus in a multi-processor mode. The `n` variable specified with this parameter pr Innovus multi-CPU usage. *Data_type*: int, optional
`-distributed_timeout seconds`	Specifies the wait time for the DP processes to start. *Data_type*: int, optional
`-extra_options options`	Specifies the additional run options. *Data_type*: string, optional
`-fix_post_process_drc`	Runs the fill clean-up utility to resolve the DRC issues, which appear post the fill steps. *Data_type*: bool, optional
`-format {stream oasis}`	Specifies the format of the flow. *Data_type*: enum, optional
`-incremental`	Runs Pegasus incremental track fill. **Note**: This parameter is used with the `-initial_working_dir` and `-working_dir` paramete *Data_type*: bool, optional
`-initial_working_dir string-initial_working_dir string`	Specifies the working directory with taking information from the previous run. *Data_type*: string, optional
`-layer_map_file streamOut_map_file`	Specifies the name of the `write_stream` layer map file. *Data_type*: string, optional
`-lib_name library_name`	Specifies the library name. *Data_type*: string, optional
`-license_distributed_continue`	Specifies that if the required number of licenses cannot be obtained in the DP processes, licenses obtained. *Data_type*: bool, optional
`-license_timeout seconds`	Specifies the wait time (in seconds) to run the software, if a license is not immediately ava *Default*: `300` *Data_type*: int, optional
`-merge list_of_external_stream_files`	Specifies the list of external stream files. *Data_type*: string, optional
`-nc resource_string`	Runs the fill process through NC with the specified resource string. *Data_type*: string, optional
`-no_structure_name`	Merges the GDSII file without loading the software database. **Note:** This parameter is used with the `-merge` parameter. *Data_type*: bool, optional
`-offset {x y}`	Specifies the location (in microns) of the design in the GDSII output file. *Data_type*: point, optional
`-output_macros`	Displays the LEF abstract information, such as LEF pin geometries and obstructions for m *Data_type*: bool, optional
`-qsub resource_string`	Runs the fill process through QSUB with the specified resource string. *Data_type*: string, optional

`-report_file`	Specifies the cell name to be changed to the file. *Data_type*: bool, optional
`-stripes` *number*	Specifies the number of stripes that divide the design. *Default*: 1 *Data_type*: int, optional
`-structure_name` *structureName*	Specifies the superstructure that consists of all the geometries that the map file writes out. *Data_type*: string, optional
`-track_fill_config_file` *file_name*	Specifies the configuration file for track fill flow.
`-turbo` *jobs_num*	Speeds up the fill process by running the specified number of parallel fill jobs. *Data_type*: int, optional
`-uniquify_cell_names`	Creates unique cell names if there are name collisions, when the software merges files. *Data_type*: bool, optional
`-units {100 | 200 | 400 | 800 | 1000 | 2000 | 4000 | 8000 | 10000 | 20000}`	
	Specifies the resolution for values in the GDSII/OASIS file. *Data_type*: enum, optional
`-working_dir` *working_directory*	Specifies the working directory. *Data_type*: string, optional

add_via_fill

```
add_via_fill
[-help]
[-area x1 y1 x2 y2]
[-layers [cut_layer_number | {list_of_cut_layer_numbers}]]
[-include_vias vias]
[-modes {all | floatingOnly | connectedOnly | {floating connectToPG connectBetweenFill}}]
[-snap]
```

Uses the values specified by the `set_metal_fill` command to determine sizes of vias to create. Via fill is inserted to get uniform cut density. Only vias that meet Minimum Area Rule requirements are used. `add_via_fill` also uses values from `set_via_fill` for the window size, step, and density values.

For information, see `set_metal_fill` and `set_via_fill`.

Note: Default values from `set_metal_fill` for `min_width` and `min_length` are biased towards the needs of `add_metal_fill` use and should be overridden with values that are more useful for via generation.

Add via fill in the following areas immediately before adding metal fill:

- Where metal fill shapes would intersect with power or ground nets on consecutive layers if no via fill was added.
- Where metal fill shapes on consecutive layers would intersect if no via fill was added.

To reduce the need for via fill, insert multiple-cut vias with the NanoRoute router. For information, see route Category Attributes.

Adding Via Fill in Multi-Threading Mode

You can add via fill in multi-threading mode by using the `set_multi_cpu_usage` command before adding the via fill.

Via Fill supports the new `set_multi_cpu_usage` command use model. For more information, see the following documents:

- Multiple-CPU Processing Commands chapter of the *Innovus Text Command Reference*.
- Accelerating the Design Process By Using Multiple-CPU Processing chapter of the *Innovus User Guide*.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `add_via_fill` parameter. For a detailed description of the command and all its parameters, use the `man` command: `man add_via_fill`
`-area x1 y1 x2 y2`	Specifies a set of coordinates within which to add via fill. For example, specify `-area 0 0 100 100`. *Default:* If you do not specify this parameter, adds via fill to the entire design area. *Data_type*: rect, optional

`-include_vias vias`	Specifies a list of vias (VIA in LEF) to be included or via rules (VIARULE GENERATE in LEF) to be used or a combination of the two. Separate the arguments with spaces. You can specify both via names and via rules.
	• The following example shows how to specify a list of via names:
	`-include_vias {VIA12 VIA23}`
	• The following example shows how to specify a list of via rules:
	`-include_vias {via1Array via2Array}`
	• The following example shows how to specify both via names and via rules:
	`-include_vias {VIA12 VIA23 via1Array via2Array}`
	Note: Non-default rule (NDR) vias can also be specified with `-include_vias`.
	Data_type: string, optional

`-layers {`*`cut_layer_number`*` | {`*`list_of_cut_layer_numbers`*`}}`	Specifies the cut layers for adding via fill. If you specify more than one layer, separate each layer with a space and enclose the list in braces or double quotation marks. To specify *cut12*, *cut23*, and *cut34*, type one of the following commands:
	`add_via_fill -layers "VIA12 VIA23 VIA34"`
	`add_via_fill -layers {VIA12 VIA23 VIA34}`
	Default: If you do not specify this parameter, adds via fill to all cut layers.
	Data_type: string, optional

`-modes {all | floatingOnly | connectedOnly | {floating connectToPG connectBetweenFill}}`	
	Specifies the connection mode for via fill. Via fill can be connected to power or ground nets (tied off), or connected between floating metal fill, or unconnected to metal fill (floating). Via fill cannot be connected to signal nets. Specify one or more of the following parameters, as required: • `all`: Enables addition of unconnected and connectToPG via fill. • `floatingOnly`: Adds unconnected via fill only. The option will be made obsolete in a future release. • `floating`: Adds unconnected via fill. It is equivalent to `-modes floatingOnly`, which will be made obsolete in a future release. • `connectToPG`: Adds connected via fill to PG. It is equivalent to `-modes connectOnly`, which will be made obsolete in a future release. • `connectBetweenFill`: Adds connected via fill between floating metal fill. • `connectedOnly`: Adds connected via fill to PG only. The option will be made obsolete in a future release. *Default:* `all` *Data_type*: string, optional
`-snap`	Snaps edges of via fill shapes and centers of cuts to a user-defined snap grid that is a multiple of the manufacturing grid. When via shapes are snapped to this grid, the edges and centers of the via shapes lie on the grid. If a grid is not specified, the software snaps the shapes to the manufacturing grid. You can define a grid using the `set_snap_grid` command. For more information, see `set_snap_grid`. *Data_type*: bool, optional

Example

The following set of command inserts unconnected via fill and connected via fill between floating metal fill in layers `VIA1` to `VIA8` and `RV` based on existing `set_metal_fill` and `set_via_fill` settings:

```
set_via_fill -layers {VIA1 VIA2 VIA3 VIA4 VIA5 VIA6 VIA7} /
           -window_size 100 100 -window_step 50 50 -min_density 0.01 /
           -max_density 30 -preferred_density 1

set_via_fill -layers {VIA8 RV} -window_size 100 100 -window_step 50 50 /
           -min_density 0.25 -max_density 30 -preferred_density 10

add_via_fill -layers {VIA1 VIA2 VIA3 VIA4 VIA5 VIA6 VIA7 VIA8 RV} /
           -modes {floating connectBetweenFill}
```

Related Information

The following sections in the Optimizing Metal Density chapter of the *Innovus User Guide*:

- Overview
- Adding Via Fill

delete_metal_fill

```
delete_metal_fill
[-help]
[-area {x1 y1 x2 y2}]
[-layers {layer_number | {list_of_layer_numbers} } ]
[-modes {all | connected_only | floating_only} ]
[-shapes {[FILLWIRE] [FILLWIREOPC}] ]
```

Removes metal and via fill in order to start over with the metal fill process. If you run this command without specifying parameters, it deletes all the metal and via fill in the design.

Note: Deleting via fill with this command might create open violations.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `delete_metal_fill` parameter. For a detailed description of the command and all its parameters, use the `man` command: `man delete_metal_fill`
`-area {x1 y1 x2 y2}`	Specifies a set of coordinates within which to delete metal and via fill. For example, specify `-area {0 0 100 100}`. *Default:* If you do not specify this parameter, deletes fill from the entire design area. *Data_type*: string, optional

`-layers {layer_number | {list_of_layer_numbers}}`	Specifies the layers from which to remove metal and via fill. If you specify more than one layer, separate each layer with a space and enclose the list in braces or double quotation marks. This parameter accepts metal layers, cut layers, or a combination.
	• If you specify a cut layer, the command removes vias that contain that cut layer only. For example:
	`delete_metal_fill -layers {VIA2 VIA3}`
	• If you specify a metal layer, metal fill shapes on that layer and vias that contain that layer are removed. For example:
	`delete_metal_fill -layers {metal2 metal3}`
	Default: If you do not specify this parameter, removes metal and via fill from all layers.
	Data_type: string, optional

`-modes {all | connected_only | floating_only}`	Specifies whether the command removes only connected fill, only floating fill, or all fill.
	Default: `all`
	Removes connected and floating metal and via fill.
	`connected_only`
	Removes only connected metal and via fill.
	`floating_only`
	Removes only floating metal and via fill.
	Data_type: enum, optional

`-shapes {[FILLWIRE] [FILLWIREOPC]}`	Specifies whether the software removes `FILLWIRE` or `FILLWIREOPC` shapes or both kinds of shapes. *Default:* `FILLWIRE FILLWIREOPC` *Data_type*: string, optional

Example

- The following command removes via fill from cut layers `VIA12`, `VIA23`, `VIA34`, `VIA45`, `VIA56`:
  ```
  delete_metal_fill -layers {VIA12 VIA23 VIA34 VIA45 VIA56}
  ```
- The following command removes only floating `FILLWIREOPC` via fill from layer metal2 and vias containing that layer:
  ```
  delete_metal_fill -area {0 0 100 100} -layers metal2 -modes floating_only /
  -shapes FILLWIREOPC
  ```

Related Information

The Optimizing Metal Density chapter of the *Innovus User Guide*:

fix_open_fill

```
fix_open_fill
[-help]
[-nets list_of_net_names]
[-area x1 y1 x2 y2]
[-remove]
```

Fixes any isolated power/ground (PG) fills. Isolated PG fills are FILL shapes attached to a power-ground (PG) net that are not connected to at least one non-FILL shape.

In designs with floating fill, the `trim_metal_fill` command is used after routing signal nets to trim fill that causes DRC violations. In designs with tied fill, i.e. metal or via fill connected to special nets, `trim_metal_fill` causes isolated fills to appear. In such cases, use the `fix_open_fill` command after `trim_metal_fill` to either change the isolated fills to floating fills or to remove them from the design.

If you choose to remove the isolated fills with `fix_open_fill -remove`, you can then run incremental `add_metal_fill` to see if any of those islands can be tied, either to the same or to another PG rail.

If you run the `fix_open_fill` command without specifying any parameters, it changes all isolated fills in the design to floating.

Parameters

`-help`	Prints a brief description that includes the type and default information for each `fix_open_fill` parameter. For a detailed description of the command and all its parameters, use the `man` command: `man fix_open_fill`
`-area x1 y1 x2 y2`	Specifies a set of coordinates within which to fix open fill issues. *Default:* If you do not specify this parameter, fixes open fills in the entire design area. *Data_type*: rect, optional
`-nets list_of_net_names`	Specifies a list of special net names to fix. *Data_type*: string, optional
`-remove`	Specifies whether to remove the open fills. If this option is not specified, `fix_open_fill` changes isolated PG fills to floating fills. *Data_type*: bool, optional

Example

The following command removes all isolated PG fills from the design:

```
fix_open_fill -remove
```

Related Information

The following sections in the Optimizing Metal Density chapter of the *Innovus User Guide*:

- Overview
- Connected and Floating Metal Fill
- Trimming Metal Fill

get_metal_fill_spacing_table

```
get_metal_fill_spacing_table
[-help]
[-layers layer_name_list]
[-fill_to_active]
[-fill_to_fill]
[-fill_to_opc]
[-opc_to_active]
[-opc_to_opc]
```

Prints the spacing table for specified or for all layers. If no option is specified, prints all spacing tables for all layers.

To specify spacing table based on existing metal fill width, use the `set_metal_fill_spacing_table` command.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `get_metal_fill_spacing_table` parameter. For a detailed description of the command and all its parameters, use the `man` command: `man get_metal_fill_spacing_table`
`-fill_to_active`	Prints the fill to active spacing table for specified or all layers. *Data_type*: bool, optional
`-fill_to_fill`	Prints the fill to fill spacing table for specified or all layers. *Data_type*: bool, optional
`-fill_to_opc`	Prints the fill to OPC spacing table for specified or all layers. *Data_type*: bool, optional
`-layers` *layer_name_list*	Prints spacing table for specified layers. *Default*: Prints spacing table for all layers *Data_type*: string, optional
`-opc_to_active`	Prints the OPC to active spacing table for specified or all layers. *Data_type*: bool, optional
`-opc_to_opc`	Prints the OPC to OPC spacing table for specified or all layers. *Data_type*: bool, optional

Example

- The following command prints the fill to active spacing table for layer `METAL2`:

  ```
  get_metal_fill_spacing_table -layers 2 -fill_to_active
  ```

  The output is as follows:

  ```
  Layer 2,
  fill_to_active {0.08 0.27}{0.14 0.36}{0.32 0.63}
  ```

- The following command prints all spacing tables for all layers:

  ```
  get_metal_fill_spacing_table
  ```

  The output is as follows:

  ```
  Layer metal2,
  fill_to_active {0.08 0.27}{0.14 0.36}{0.32 0.63}
  Layer metal3,
  fill_to_fill {0.08 0.2}{0.14 0.3}{0.32 0.6}
  Layer metal3,
  opc_to_active {0.05 0.08}{0.08 0.1}{0.1 0.12}
  ```

Related Information

The following sections in the Optimizing Metal Density chapter of the *Innovus User Guide*:

- Specifying Metal Fill Spacing Table
- Trimming Metal Fill

get_pg_fill_config

```
get_pg_fill_config
[-help]
[-attach_instance_name]
[-attach_net_name]
[-attach_net_prop]
[-core_license]
[-die_area_as_boundary]
[-distributed]
[-distributed_dir]
[-distributed_timeout]
[-extra_options]
[-format]
[-layer_map_file]
[-license_distributed_continue]
[-license_timeout]
[-merge]
[-no_structure_name]
[-offset]
[-output_macros]
[-pg_fill_config_file]
[-pg_hookup_flow]
[-report_file]
[-stripes]
[-structure_name]
[-uniquify_cell_names]
[-units]
[-working_dir]
```

Prints the parameters of the `add_pg_fill` command.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `get_pg_fill_config` parameter. For a detailed description of the command and all its parameters, use the `man` command: `man get_pg_fill_config`
`-attach_instance_name`	Prints the instance name to be attached. *Data_type*: bool, optional
`-attach_net_name`	Prints the net name to be attached. *Data_type*: bool, optional
`-attach_net_prop`	Prints the net property to be attached. *Data_type*: bool, optional
`-core_license`	Prints an exact combination of licenses to be used. *Data_type*: bool, optional
`-die_area_as_boundary`	Prints the die area as a polygon or a boundary. *Data_type*: bool, optional
`-distributed`	Runs Pegasus in a multi-processor mode. The *n* variable specified with this parameter presents the number of processors. By default, this value is taken from the Innovus multi-CPU usage. *Data_type*: bool, optional
`-distributed_dir`	Prints the name of the distributed directory. *Data_type*: bool, optional

`-distributed_timeout`	Prints the wait time for the distributed processes to start. *Data_type*: bool, optional
`-extra_options`	Prints the additional run options. *Data_type*: bool, optional
`-format`	Prints the format of the flow. *Data_type*: bool, optional
`-layer_map_file`	Prints the name of the `write_stream` layer map file. *Data_type*: bool, optional
`-lib_name`	Prints the library name. *Data_type*: bool, optional
`-license_distributed_continue`	Specifies that if the required number of licenses cannot be obtained in the distributed processes, the software continues with fewer CPUs based on the number of licenses obtained. *Data_type*: bool, optional
`-license_timeout`	Prints the wait time (in seconds) to run the software, if a license is not immediately available. *Default*: `300` *Data_type*: bool, optional
`-merge`	Prints the list of external stream files. *Data_type*: bool, optional
`-no_structure_name`	Merges the GDSII file without loading the software database. **Note:** This parameter is used with the `-merge` parameter. *Data_type*: bool, optional
`-offset`	Prints the location (in microns) of the design in the GDSII output file. *Data_type*: bool, optional
`-output_macros`	Prints the LEF abstract information, such as LEF pin geometries and obstructions for macros. *Data_type*: bool, optional
`-pg_fill_config_file`	Prints the configuration file for the PG fill flow. Any overrides to the default behavior are written in this file. *Data_type*: bool, optional
`-pg_hookup_flow`	Prints the type of fills to be generated. *Default*: `PG_ONLY` *Data_type*: bool, optional
`-report_file`	Prints the cell name to be changed to the file. *Data_type*: bool, optional
`-stripes`	Prints the number of stripes that divide the design. *Default*: 1 *Data_type*: bool, optional
`-structure_name`	Prints the superstructure that consists of all the geometries that the map file writes out. *Data_type*: bool, optional
`-uniquify_cell_names`	Creates unique cell names if there are name collisions, when the software merges files. *Data_type*: bool, optional
`-units`	Prints the resolution for values in the GDSII/OASIS file. *Data_type*: bool, optional

`-working_dir` *working_directory*	Prints the working directory. *Data_type*: bool, optional

get_track_fill_config

```
get_track_fill_config
[-help]
[-attach_instance_name]
[-attach_net_name]
[-attach_net_prop]
[-core_license]
[-die_area_as_boundary]
[-distributed]
[-distributed_dir]
[-distributed_timeout]
[-extra_options]
[-format]
[-layer_map_file]
[-license_distributed_continue]
[-license_timeout]
[-merge]
[-no_structure_name]
[-offset]
[-output_macros]
[-report_file]
[-stripes]
[-structure_name]
[-track_fill_config_file]
[-uniquify_cell_names]
[-units]
[-working_dir]
```

Prints the parameters of the `add_track_fill` command.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `get_track_fill_config` parameter. For a detailed description of the command and all its parameters, use the `man` command: `man get_track_fill_config`
`-attach_instance_name`	Prints the instance name to be attached. *Data_type*: bool, optional
`-attach_net_name`	Prints the net name to be attached. *Data_type*: bool, optional
`-attach_net_prop`	Prints the net property to be attached. *Data_type*: bool, optional
`-core_license`	Prints an exact combination of licenses to be used. *Data_type*: bool, optional
`-die_area_as_boundary`	Prints the die area as a polygon or a boundary. *Data_type*: bool, optional
`-distributed`	Runs Pegasus in a multi-processor mode. The *n* variable specified with this parameter presents the number of processors. By default, this value is taken from the Innovus multi-CPU usage. *Data_type*: bool, optional
`-distributed_dir`	Prints the name of the distributed directory. *Data_type*: bool, optional
`-distributed_timeout`	Prints the wait time for the distributed processes to start. *Data_type*: bool, optional

`-extra_options`	Prints the additional run options. *Data_type*: bool, optional
`-format`	Prints the format of the flow. *Data_type*: bool, optional
`-layer_map_file`	Prints the name of the `write_stream` layer map file. *Data_type*: bool, optional
`-lib_name`	Prints the library name. *Data_type*: bool, optional
`-license_distributed_continue`	Specifies that if the required number of licenses cannot be obtained in the distributed processes, the software continues with fewer CPUs based on the number of licenses obtained. *Data_type*: bool, optional
`-license_timeout`	Prints the wait time (in seconds) to run the software, if a license is not immediately available. *Default*: `300` *Data_type*: bool, optional
`-merge`	Prints the list of external stream files. *Data_type*: bool, optional
`-no_structure_name`	Merges the GDSII file without loading the software database. **Note:** This parameter is used with the `-merge` parameter. *Data_type*: bool, optional
`-offset`	Prints the location (in microns) of the design in the GDSII output file. *Data_type*: bool, optional
`-output_macros`	Prints the LEF abstract information, such as LEF pin geometries and obstructions for macros. *Data_type*: bool, optional
`-report_file`	Prints the cell name to be changed to the file. *Data_type*: bool, optional
`-stripes`	Prints the number of stripes that divide the design. *Default*: 1 *Data_type*: bool, optional
`-structure_name`	Prints the superstructure that consists of all the geometries that the map file writes out. *Data_type*: bool, optional
`-track_fill_config_file` *file_name*	Specifies the configuration file for track fill flow. *Data_type*: bool, optional
`-uniquify_cell_names`	Creates unique cell names if there are name collisions, when the software merges files. *Data_type*: bool, optional
`-units`	Prints the resolution for values in the GDSII/OASIS file. *Data_type*: bool, optional
`-working_dir` *working_directory*	Prints the working directory. *Data_type*: bool, optional

report_metal_fill

```
report_metal_fill
[-help]
[-out_file file_name]
[-fill_area]
[-fill_length]
[-mesh]
```

Generates a report on metal fill to summarize the number and area of fill shapes on a per-net and per-layer basis. Use the generated report to tune the `add_metal_fill` flow parameters to achieve the desired distribution across multiple nets and layers as well as to minimize floating fill.

Before generating a metal fill report, you should have:

- Imported the design
- Completed detailed routing
- Added metal fill

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `report_metal_fill` parameter. For a detailed description of the command and all its parameters, use: `man report_metal_fill`
`-fill_area`	Reports area information for metal fill for each layer. *Data_type*: bool, optional
`-fill_length`	Reports length information for metal fill for each layer. *Data_type*: bool, optional
`-mesh`	Reports mesh information for metal fill. *Data_type*: bool, optional
`-out_file file_name`	Writes the report to a file with the specified name. *Default*: *top_cell_name*`_metal_fill_report.rpt` *Data_type*: string, optional

Example

- The following command writes a metal fill report to the dtmf_metal_fill.rpt file.

```
report_metal_fill -out_file dtmf_metal_fill.rpt
```

The output is as follows:

```
#############################################################

# Generated by: Cadence Innovus 15.10-s076_1

# OS: Linux x86_64(Host ID lnx-opsdev)

# Generated on: Tue Feb 22 15:46:26 2011

# Design: dtmf_chip

# Command: report_metal_fill -out_file dtmf_metal_fill.rpt

#############################################################

Metal Fill Summary

<numFillShapes>=<fillArea>squm(<percentTotal>)

 |floating |total

-----------------------------------------------------

 metal1| 9633= 138916.93 (100.00%) | 9633= 138916.93

 metal2|11619= 143084.45 (100.00%) |11619= 143084.45

 metal3| 9432= 121945.18 (100.00%) | 9432= 121945.18

 metal4|10840= 133624.91 (100.00%) |10840= 133624.91

 metal5|12447= 177853.14 (100.00%) |12447= 177853.14

 metal6|12687= 177747.55 (100.00%) |12687= 177747.55

 metal7|12714= 187099.41 (100.00%) |12714= 187099.41

 metal8|20403= 319443.23 (100.00%) |20403= 319443.23

-----------------------------------------------------

sum|99775=1399714.80 (100.00%) |99775=1399714.80
```

- The following command reports the length information for metal fill:

```
report_metal_fill -fill_length
```

The output is as follows:

```
#############################################################

# Generated by: Cadence Innovus 15.10-s076_1
```

```
# OS: Linux x86_64(Host ID lnx-opsdev)

# Generated on: Tue Feb 22 15:56:35 2011

# Design: dtmf_chip

# Command: report_metal_fill -fill_length

#############################################################

Metal Fill Summary

<numFillShapes>=<fillLength>um(<percentTotal>)

 |floating |total

-----------------------------------------------------

 metal1| 9633= 71556.12 (100.00%) | 9633= 71556.12

 metal2|11619= 75602.62 (100.00%) |11619= 75602.62

 metal3| 9432= 64887.96 (100.00%) | 9432= 64887.96

 metal4|10840= 70622.95 (100.00%) |10840= 70622.95

 metal5|12447= 92946.43 (100.00%) |12447= 92946.43

 metal6|12687= 91974.32 (100.00%) |12687= 91974.32

 metal7|12714= 97591.75 (100.00%) |12714= 97591.75

 metal8|20403=164491.39 (100.00%) |20403=164491.39

-----------------------------------------------------

sum|99775=729673.56 (100.00%) |99775=729673.56
```

Related Information

The Optimizing Metal Density chapter of the *Innovus User Guide*.

set_metal_fill

```
set_metal_fill
[-help]
[-layer {layerNumber} | {list_of_layer_numbers}]
[-active_spacing spacing]
[-opc_active_spacing spacing]
[-border_spacing spacing]
[-decrement value]
[-diagonal_offset x y]
[-external_density percent]
[-gap_spacing spacing]
[-honor_lef_value]
[-iteration_name name]
[-max_density percent]
[-max_length length]
[-max_union_density_to_above_layer float]
[-max_width width]
[-min_density percent]
[-min_length length]
[-min_union_density_to_above_layer float]
[-min_width width]
[-opc | -regular]
[-preferred_density percent]
[-window_size xSize ySize]
[-window_step xStep yStep]
```

Specifies values for parameters that the `add_metal_fill`, `trim_metal_fill`, and `check_metal_density` commands use for metal fill size, spacing, verification, and metal density. You can also specify some of these parameters in the Layer (Routing) section of the LEF file - they are identified in the "Parameters" section below.

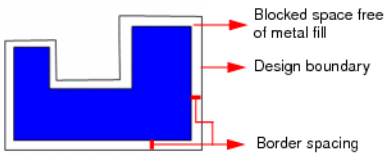
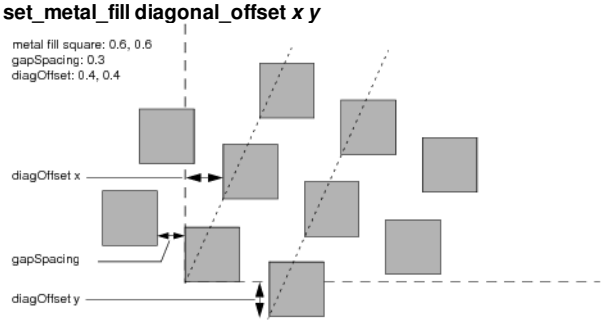
If you want your metal fill settings to be saved in the .init file when saving the design, you must set the write_db_include_metal_fill_rules global to `1` before saving the design. You can easily reload your settings by restoring the saved design

Note: Check the chip manufacturer's DRC manual specifications for metal fill (sometimes called dummy fill) for the correct values if they are not specified in the LEF file.

- For more information on LEF, see the LEF Syntax chapter in the *LEF/DEF Language Reference*.
- For more information on the metal fill commands, see the `add_metal_fill`, `trim_metal_fill`, and `check_metal_density` commands.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `set_metal_fill` parameter. For a detailed description of the command and all its parameters, use the `man` command: `man set_metal_fill`
`-active_spacing spacing`	Specifies the minimum distance in microns between a metal fill shape and any other type of object in the design. For example, `set_metal_fill -active_spacing 0.6`. **Note:** The `set_metal_fill -active_spacing` settings are used for creating regular `FILLWIRE` shapes. For `FILLWIREOPC` shapes, design rules are used. LEF equivalent: `FILLACTIVESPACING` *Default:* `FILLACTIVESPACING` in the LEF file; if not specified, `0.6` for thin layers (less than 0.24), `0.8` for thick layers *Data_type*: float, optional

`-border_spacing` *spacing*	Specifies the space in microns along the design border that needs to be kept free of metal fill. This parameter is useful if you want open space next to the place and route border. If you have set border spacing, the `add_metal_fill` and `check_metal_density` commands consider the density in the specified border space area as 0.  *Data_type*: float, optional
`-decrement` *value*	Specifies the number of microns the software decreases the size of each successively smaller piece of metal fill or via fill until it reaches the size specified by the `-min_width` parameter. The software uses the maximum fill size specified unless it is impossible to fit a piece of fill into a particular area. The software then attempts to use successively smaller pieces of fill until the minimum length value is reached. This parameter allows you to control how many different sizes of metal fill or via fill can appear within the design. For example, if the value of `-max_width` is `8`, the value of `-min_width` is `4`, and the value of `-decrement` is `2`, the software can add fill shapes with widths of `8`, `6`, and `4` microns. The software tries to add fill shapes with the width specified by `-max_width`. If a shape of that width does not fit in the available space without creating a violation, the software decreases the width by the amount specified by the `-decrement` parameter. It continues to decrease the width until it reaches the width specified by the `-min_width` parameter. To increase flexibility in adding metal or via fill, decrease the value of this parameter. For example, if the value of `-decrement` is `1`, metal fill with widths of `8`, `7`, `6`, `5`, and `4` microns can be added. *Default:* `WIDTH` value specified in the `Layer` (Routing) statement of the LEF file *Data_type*: float, optional
`-diagonal_offset` *x y*	Specifies the offset for staggering metal fill. To enable this parameter, `add_metal_fill -stagger diag` must be specified. *x* Specifies the horizontal offset. *y* Specifies the vertical offset. **set_metal_fill diagonal_offset *x y***  *Data_type*: point, optional

`-external_density` *percent*	Specifies a percentage value for the external metal density for a layer when `add_metal_fill` windows lie partially outside of the die area. For each layer, a different external density value can be specified. Metal fill uses the external density value in the estimation of layer density outside of the blocks being developed. *Data_type*: float, optional
`-gap_spacing` *spacing*	Specifies the minimum distance in microns between one metal fill shape and another metal fill shape. For example, `set_metal_fill -gap_spacing 0.4`. *Default:* `0.4` for thin layers (less than 0.24 width/minWidth), `0.8` for thick layers *Data_type*: float, optional
`-honor_lef_value`	Specifies that the LEF values `MINIMUMDENSITY, MAXIMUMDENSITY, DENSITYCHECKWINDOW,` and`DENSITYCHECKSTEP` should be used to overwrite the `min_density, max_density, window_size` and `window_step` parameters of `set_metal_fill` if the values specified for these parameters are not valid. If the `-layer` option is used with `set_metal_fill`, only the specified layers are changed to the LEF values. **Note:** If one of the keywords`-MINIMUMDENSITY, MAXIMUMDENSITY, DENSITYCHECKWINDOW,` or`DENSITYCHECKSTEP-` is not defined in the LEF, an error message like the following is displayed: `ERROR: DENSITYCHECKWINDOW of MetalX is not defined in the LEF.` *Data_type*: bool, optional

`-iteration_name` *name*	Specifies a name for a set of values for `set_metal_fill` parameters that the software stores and uses during a later `add_metal_fill` run. Use `add_metal_fill -iteration_name_list` to add the metal fill using the stored set of parameters.
	The window size and step must be the same for all iterations of a specific layer. For example, the following specifications are not allowed because the values are not consistent:
	`set_metal_fill -iteration_name step1 -layer 1 \`
	`  -min_density 20 -window_size 100 100 -window_step 50 50`
	`set_metal_fill -iteration_name step1 -layer 1 \`
	`  -min_density 25 -window_size 50 50 -window_step 25 25`
	Important Specifying `set_metal_fill -iteration_name default` is the same as running `set_metal_fill` without specifying the `-iteration_name` parameter.
	`set_metal_fill -iteration_name default` is expected before `verify_metal_denstiy`. `verify_metal_denstiy` honors only `default iteration_name`. If `default iteration_name` is not set, `default` value will be used in `verify_metal_denstiy`.
	Default: "`default`"
	Data_type: string, optional

`-layer {`*layerNumber*`|`
`{`*list_of_layer_numbers*`}`
	Specifies the layers to which the parameters of the `set_metal_fill` command apply. If you specify more than one layer, separate each layer with a space and enclose the list in braces or double quotation marks.
	To specify metal layers Metal1, Metal2, and Metal3, type one of the following commands:
	`set_metal_fill -layer "METAL1 METAL2 METAL3"`
	`set_metal_fill -layer {METAL1 METAL2 METAL3}`
	Data_type: string, optional

`-max_density` *percent*	Specifies a percentage value for the maximum metal density allowed. LEF equivalent: `MAXIMUMDENSITY` *Default:* `MAXIMUMDENSITY`. If this value is not defined, the default is `80`. **Note:** This parameter only overrides a value in the LEF file if it is more restrictive than the LEF value. For example, if the value specified in the LEF file is 80, and you specify `-max_density 70`, metal fill uses 70 percent as the maximum density. *Data_type*: float, optional

`-max_length` *length*	Specifies the maximum length of the metal fill shapes, in microns. Metal fill stops at the chip boundary if the specified maximum length would cause it to extend outside the boundary. *Default:* `10.0` *Data_type*: float, optional
`-max_union_density_to_above_layer` *float*	Specifies the maximum allowed sum union density of the current layer and its top layer window. This avoids the union density violations found while verifying density results of trim metal fill flow using PVS or Pegasus. *Default:* `0` *Data_type*: float, optional
`-max_width` *width*	Specifies the maximum width of metal fill shapes, in microns. Metal fill stops at the chip boundary if the specified maximum length would cause it to extend outside the boundary. *Default:* Maximum routing width defined in the LEF file; if not specified, `2.0` LEF equivalent: `MAXWIDTH` **Note:** This parameter only overrides a value in the LEF file if it is more restrictive than the LEF value. *Data_type*: float, optional
`-min_density` *percent*	Specifies a percentage value for the minimum metal density allowed in the design. LEF equivalent: `MINIMUMDENSITY` *Default:* `MINIMUMDENSITY`. If this value is not defined, the default is `20`. **Note:** This parameter only overrides a value in the LEF file if it is more restrictive than the LEF value. For example, if the value specified in the LEF file is 20, and you specify `-min_density 25`, metal fill uses 25 percent as the minimum density. *Data_type*: float, optional
`-min_length` *length*	Specifies the minimum length of metal fill shapes, in microns. *Default:* `1.0` *Data_type*: float, optional
`-min_union_density_to_above_layer` *float*	Specifies the minimum allowed sum union density of the current layer and its top layer window. This avoids the union density violations found while verifying density results of trim metal fill flow using PVS or Pegasus. *Default:* `200` *Data_type*: float, optional
`-min_width` *width*	Specifies the minimum width of metal fill shapes, in microns. *Default:* `0.4` for thin layers (less than 0.24) and `0.8` for thick layers. **Note:** This parameter only overrides a value in the LEF file if it is more restrictive than the LEF value. *Data_type*: float, optional
`-opc`	Specifies metal fill that will go through the optical proximity correction (OPC) process. This fill can be smaller than non-OPC metal fill and is governed by the same design rules as regular wires, instead of the rules that govern non-OPC metal fill. • OPC metal fill that is floating (unconnected) is included in the Fills statement of DEF 5.7. • OPC metal fill that is tied off (connected) is included in the Special Wiring statement of DEF 5.7. • The GDSII Stream/OASIS map file identifies the following object names for OPC and non-OPC metal fill: `FILL`, `FILLOPC`, `VIA`, `VIAFILL`. **Note:** `-opc` and `-regular` cannot be specified at the same time. *Data_type*: bool, optional
`-opc_active_spacing` *spacing*	Specifies the minimum distance in microns between the OPC metal fill shapes (FILLWIREOPC) and main patterns. **Note:** If this parameter is not specified, the minimum DRC spacing is used by default. *Data_type*: float, optional

`-preferred_density` *percent*	Specifies a percentage value for the preferred metal density for the layer. When the metal density in a window is less than the minimum metal fill density value, adds metal fill to achieve a density slightly above the preferred density if possible. In addition, metal fill uses the preferred density value in the estimation of `MACRO` density for layers that have most of the `MACRO` covered by `OBS` shapes and for DEF `BLOCKAGES` without both `+ FILLS` and `+ PUSHDOWN`. Metal fill uses the following scheme for the treatment of DEF blockages with `+ FILLS` or `+ PUSHDOWN`: • DEF `BLOCKAGES` with `+ PUSHDOWN` = 100 percent density (the same as real metal segments) • DEF `BLOCKAGES` with `+ FILLS` = 0 percent density Cadence recommends modelling LEF `MACRO`s by using the `DENSITY` statement, with density windows that are smaller than the metal fill windows. For example, if the metal fill windows are 100 x 100, use 25 x 25 or smaller values for the density rectangles. For more information, see the "Adding Metal Fill Over Macros" section in the Optimizing Metal Density chapter of the *Innovus User Guide.* *Default:* `35` *Data_type*: float, optional

`-regular`	Specifies that regular metal fill shape is to be used. Use this option to switch to inserting regular metal fill without restarting the session after you have been adding OPC metal fill.
	`set_metal_fill -opc`
	`add_metal_fill`
	`#Inserts opc metal fill`
	`set_metal_fill -regular`
	`add_metal_fill`
	`#Inserts regular metal fill`
	Data_type: bool, optional

`-window_size` *xSize ySize*	Specifies the x and y dimensions in microns of the area `set_metal_fill` uses to examine the metal density. The software assesses the metal density in each window, moving the window iteratively through the design. For example, `set_metal_fill -window_size 100 100`. LEF equivalent: `DENSITYCHECKWINDOW` *Default:* `DENSITYCHECKWINDOW`. If this value is not defined, the default is 100 microns in each dimension. *Data_type*: point, optional
`-window_step` *xStep yStep*	Specifies the x and y step distance in microns the window moves for each iteration of metal density analysis. For example, `set_metal_fill -window_step 50 50`. LEF equivalent: `DENSITYCHECKSTEP` *Default:* `DENSITYCHECKSTEP`. If this value is not defined, the default is 50 microns in each dimension. *Data_type*: point, optional

Example

- The following command sets window size and step settings for layer `METAL1`:

  ```
  set_metal_fill -layer METAL1 -window_size 200 200 -window_step 100 100
  ```

  These parameters are applied when you use the `add_metal_fill` command.

- The following command specifies an iteration `step1` with various settings for inserting metal fill:

  ```
  set_metal_fill -layer {1 2 3 4 5 6 7 8} -iteration_name step1 /
                 -gap_spacing 0.140 -min_width 1 -decrement 0.050 /
                 -min_length 2 -max_width 5 -max_length 20 -active_spacing 0.8 /
                 -window_size 50 50 -window_step 25 25 -min_density 15 /
                 -max_density 80 -preferred_density 35 -diagonal_offset 1 1
  ```

Related Information

- The following sections in the Optimizing Metal Density chapter of the *Innovus User Guide* document:
  - Overview
  - Connected and Floating Metal Fill
  - Specifying Metal Fill Parameters
  - Timing-Aware Examples
  - Specifying the Active Spacing Value
  - Adding Via Fill
- The 'GDSII Stream or OASIS Map File Format' section in the 'Importing and Exporting Designs' chapter in the Innovus User Guide document
- The 'Macro Density' and 'Blockages' sections in the LEF Syntax chapter in the *LEF/DEF Language Reference* document

set_metal_fill_spacing_table

```
set_metal_fill_spacing_table
[-help]
[-layer layer_name_list]
[-fill_to_active {{width1 spacing1}{width2 spacing2}...}]
[-fill_to_fill {{width1 spacing1}{width2 spacing2}...}]
[-fill_to_opc {{width1 spacing1}{width2 spacing2}...}]
[-opc_to_active {{width1 spacing1}{width2 spacing2}...}]
[-opc_to_opc {{width1 spacing1}{width2 spacing2}...}]
[-reset]
```

Specifies the spacing table used by `trim_metal_fill` based on existing metal fill width. The spacing table includes the following spacing values:

- Fill to active spacing
- Fill to fill spacing
- Fill to OPC spacing
- OPC to OPC spacing
- OPC to active spacing

As `trim_metal_fill` honors the spacing table, you can then run `trim_metal_fill` to trim metal fill and get expected spacing. When required, you can use the `get_metal_fill_spacing_table` command to print the spacing table.

Note: If no spacing table is specified, `trim_metal_fill` honors the `set_metal_fill` settings for `-active_spacing` and `-gap_spacing`. If `set_metal_fill` values are also not specified, `trim_metal_fill` uses the default settings.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `set_metal_fill_spacing_table` parameter. For a detailed description of the command and all its parameters, use the `man` command: `man set_metal_fill_spacing_table`
`-layer` *layer_name_list*	Specifies the metal layers for the setting. Metal layers can be specified as numbers (`2`) or in the LEF format (`METAL2`) or as metal`2`. *Data_type*: string, optional
`-fill_to_active {{`*width1 spacing1*`}{`*width2 spacing2*`}...}`	Specifies a spacing table for fill to active spacing for trimming metal fill. *Data_type*: string, optional
`-fill_to_fill {{`*width1 spacing1*`}{`*width2 spacing2*`}...}`	Specifies a spacing table for fill to fill spacing for trimming metal fill. *Data_type*: string, optional
`-fill_to_opc {{`*width1 spacing1*`}{`*width2 spacing2*`}...}`	Specifies a spacing table for fill to OPC spacing for trimming metal fill. *Data_type*: string, optional
`-opc_to_active {{`*width1 spacing1*`}{`*width2 spacing2*`}...}`	Specifies a spacing table for OPC to active spacing for trimming metal fill. *Data_type*: string, optional
`-opc_to_opc {{`*width1 spacing1*`}{`*width2 spacing2*`}...}`	Specifies a spacing table for OPC to OPC spacing for trimming metal fill. *Data_type*: string, optional
`-reset`	Resets the spacing tables. *Data_type*: bool, optional

Example

The following set of commands specify the spacing table for fill to active spacing as given below and trim metal fill accordingly:

Layer	min_width (=>) max_width (<=) active_spacing (>=)
2	0.32 ∞ 0.63
2	0.14 0.32-1MFG 0.36
2	0.08 0.14-1MFG 0.27

```
set_metal_fill_spacing_table -layer {2} -fill_to_active {{0.08 0.27}{0.14 0.36}{0.32 0.63}}
trim_metal_fill -layer 2
```

The software checks the existing metal fill with the specified spacing table. If no violations are detected, existing metal fill is retained. If the existing metal fills have `active_spacing` violations with other features, `trim_metal_fill` uses the `fill_to_active` spacing table to trim metal fill.

Sample output is as follows:

```
*** START TRIM METALFILL *** (CPU Time: 0:00:00.0 MEM: 558.359M)
 Number of metal fills with spacing or/and short violations:4920
 Total number of deleted metal fills: 4920
 Total number of added metal fills: 4779
 (CPU Time: 0:00:01.2 MEM: 558.359M)
 *** END OF TRIM METALFILL ***
```

Related Information

The following sections in the Optimizing Metal Density chapter of the *Innovus User Guide* document:

- Specifying Metal Fill Spacing Table
- Trimming Metal Fill

set_pg_fill_config

```
set_pg_fill_config
[-help]
[-attach_instance_name attr_num]
[-attach_net_name attr_num]
[-attach_net_prop {{prop_name attr_num} ... }]
[-core_license license_combination]
[-die_area_as_boundary]
[-distributed n]
[-distributed_timeout seconds]
[-extra_options options]
[-format {stream oasis}]
[-layer_map_file streamOut_map_file]
[-lib_name library_name]
[-license_distributed_continue]
[-license_timeout seconds]
[-merge list_of_external_stream_files]
[-no_structure_name]
[-offset {x y}]
[-output_macros]
[-pg_fill_config_file file_name]
[-pg_hookup_flow {ALL | PG_ONLY | FLOAT_ONLY}]
[-report_file]
[-reset {parameter_name | all}]
[-stripes number]
[-structure_name structureName]
[-uniquify_cell_names]
[-units {100 | 200 | 400 | 800 | 1000 | 2000 | 4000 | 8000 | 10000 | 20000}]
[-working_dir working_directory]
```

Sets the parameters of the `add_pg_fill` command.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `set_pg_fill_` For a detailed description of the command and all its parameters, use the `man` command: `man set_pg_fill_config`
`-attach_instance_name attr_num`	Specifies the instance name to be attached. *Data_type*: int, optional
`-attach_net_name attr_num`	Specifies the net name to be attached. *Data_type*: int, optional
`-attach_net_prop {{prop_name attr_num} ... }`	Specifies the net property to be attached. *Data_type*: string, optional
`-core_license license_combination`	Sets an exact combination of licenses to be used. *Data_type*: string, optional
`-die_area_as_boundary`	Writes the die area as a polygon or a boundary. When this parameter is not specified, the `write` *Data_type*: bool, optional
`-distributed n`	Runs Pegasus in a multi-processor mode. The `n` variable specified with this parameter presents Innovus multi-CPU usage. *Data_type*: int, optional
`-distributed_timeout seconds`	Specifies the wait time for the DP processes to start. *Data_type*: int, optional

`-extra_options` *options*	Specifies the additional run options. *Data_type*: string, optional
`-format {stream oasis}`	Specifies the format of the flow. *Data_type*: enum, optional
`-layer_map_file` *streamOut_map_file*	Specifies the name of the `write_stream` layer map file. *Data_type*: string, optional
`-lib_name` *library_name*	Specifies the library name. *Data_type*: string, optional
`-license_distributed_continue`	Specifies that if the required number of licenses cannot be obtained in the DP processes, the so licenses obtained. *Data_type*: bool, optional
`-license_timeout` *seconds*	Specifies the wait time (in seconds) to run the software, if a license is not immediately available *Default*: `300` *Data_type*: int, optional
`-merge` *list_of_external_stream_files*	Specifies the list of external stream files. *Data_type*: string, optional
`-no_structure_name`	Merges the GDSII file without loading the software database. **Note:** This parameter is used with the `-merge` parameter. *Data_type*: bool, optional
`-offset {`*x y*`}`	Specifies the location (in microns) of the design in the GDSII output file. *Data_type*: point, optional
`-output_macros`	Displays the LEF abstract information, such as LEF pin geometries and obstructions for macros *Data_type*: bool, optional
`-pg_fill_config_file` *file_name*	Specifies the configuration file for the PG fill flow. Any overrides to the default behavior are writt *Data_type*: string, optional
`-pg_hookup_flow {ALL | PG_ONLY | FLOAT_ONLY}`	Specifies the type of fills to be generated. *Default*: `PG_ONLY` *Data_type*: enum, optional
`-report_file`	Specifies the cell name to be changed to the file. *Data_type*: bool, optional
`-reset {`*parameter_name* `|` *all*`}`	Resets a specified parameter or all parameters to the default value.
`-stripes` *number*	Specifies the number of stripes that divide the design. *Default*: 1 *Data_type*: int, optional
`-structure_name` *structureName*	Specifies the superstructure that consists of all the geometries that the map file writes out. *Data_type*: string, optional
`-uniquify_cell_names`	Creates unique cell names if there are name collisions, when the software merges files. *Data_type*: bool, optional
`-units {100 | 200 | 400 | 800 | 1000 | 2000 | 4000 | 8000 | 10000 | 20000}`	
	Specifies the resolution for values in the GDSII/OASIS file. *Data_type*: enum, optional

`-working_dir working_directory`	Specifies the working directory. *Data_type*: string, optional

set_track_fill_config

```
set_track_fill_config
[-help]
[-attach_instance_name attr_num]
[-attach_net_name attr_num]
[-attach_net_prop {{prop_name attr_num} ... }]
[-core_license license_combination]
[-die_area_as_boundary]
[-distributed n]
[-distributed_timeout seconds]
[-extra_options options]
[-format {stream oasis}]
[-layer_map_file streamOut_map_file]
[-lib_name library_name]
[-license_distributed_continue]
[-license_timeout seconds]
[-merge list_of_external_stream_files]
[-no_structure_name]
[-offset {x y}]
[-output_macros]
[-report_file]
[-reset {parameter_name | all}]
[-stripes number]
[-structure_name structureName]
[-track_fill_config_file file_name]
[-uniquify_cell_names]
[-units {100 | 200 | 400 | 800 | 1000 | 2000 | 4000 | 8000 | 10000 | 20000}]
[-working_dir working_directory]
```

Sets the parameters of the `add_track_fill` command.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `set_track_fi` For a detailed description of the command and all its parameters, use the `man` command: `man set_track_fill_config`
`-attach_instance_name attr_num`	Specifies the instance name to be attached. *Data_type*: int, optional
`-attach_net_name attr_num`	Specifies the net name to be attached. *Data_type*: int, optional
`-attach_net_prop {{prop_name attr_num} ... }`	Specifies the net property to be attached. *Data_type*: string, optional
`-core_license license_combination`	Sets an exact combination of licenses to be used. *Data_type*: string, optional
`-die_area_as_boundary`	Writes the die area as a polygon or a boundary. When this parameter is not specified, the `write` *Data_type*: bool, optional
`-distributed n`	Runs Pegasus in a multi-processor mode. The `n` variable specified with this parameter presents Innovus multi-CPU usage. *Data_type*: int, optional
`-distributed_timeout seconds`	Specifies the wait time for the DP processes to start. *Data_type*: int, optional
`-extra_options options`	Specifies the additional run options. *Data_type*: string, optional

`-format {stream oasis}`	Specifies the format of the flow. *Data_type*: enum, optional
`-layer_map_file` *streamOut_map_file*	Specifies the name of the `write_stream` layer map file. *Data_type*: string, optional
`-lib_name` *library_name*	Specifies the library name. *Data_type*: string, optional
`-license_distributed_continue`	Specifies that if the required number of licenses cannot be obtained in the DP processes, the s licenses obtained. *Data_type*: bool, optional
`-license_timeout` *seconds*	Specifies the wait time (in seconds) to run the software, if a license is not immediately available *Default*: `300` *Data_type*: int, optional
`-merge` *list_of_external_stream_files*	Specifies the list of external stream files. *Data_type*: string, optional
`-no_structure_name`	Merges the GDSII file without loading the software database. **Note:** This parameter is used with the `-merge` parameter. *Data_type*: bool, optional
`-offset {`*x y*`}`	Specifies the location (in microns) of the design in the GDSII output file. *Data_type*: point, optional
`-output_macros`	Displays the LEF abstract information, such as LEF pin geometries and obstructions for macros *Data_type*: bool, optional
`-report_file`	Specifies the cell name to be changed to the file. *Data_type*: bool, optional
`-reset {`*parameter_name* `|` *all*`}`	Resets a specified parameter or all parameters to the default value.
`-stripes` *number*	Specifies the number of stripes that divide the design. *Default*: 1 *Data_type*: int, optional
`-structure_name` *structureName*	Specifies the superstructure that consists of all the geometries that the map file writes out. *Data_type*: string, optional
`-track_fill_config_file` *file_name*	Specifies the configuration file for track fill flow. *Data_type*: string, optional
`-uniquify_cell_names`	Creates unique cell names if there are name collisions, when the software merges files. *Data_type*: bool, optional
`-units {100 | 200 | 400 | 800 | 1000 | 2000 | 4000 | 8000 | 10000 | 20000}`	
	Specifies the resolution for values in the GDSII/OASIS file. *Data_type*: enum, optional
`-working_dir` *working_directory*	Specifies the working directory. *Data_type*: string, optional

set_via_fill

```
set_via_fill
[-help]
[-layer {layer_number | {list_of_layer_numbers}}]
[-window_size xSize ySize]
[-window_step xStep yStep]
[-min_density percent]
[-max_density percent]
[-preferred_density percent]
```

Specifies window and density values for inserting via fill. Use this command before running `add_via_fill`.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `set_via_fill` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man set_via_fill`.

`-layer layer_number | {list_of_layer_numbers}`	Specifies the cut layers to which the parameters of the `set_via_fill` command apply. If you specify more than one layer, separate each layer with a space and enclose the list in braces or double quotation marks. To specify *cut12*, *cut23*, and *cut34*, type one of the following commands:
	`add_via_fill -layer "Via12 Via23 Via34"`
	`add_via_fill -layer {Via12 Via23 Via34}`
	Data_type: string, optional

`-max_density percent`	Specifies a percentage value for the maximum cut density. For example, `set_via_fill -max_density 30`. *Default:* `30` *Data_type*: float, optional
`-min_density percent`	Specifies a percentage value for the minimum cut density. For example, `set_via_fill -min_density 0.01`. *Default:* area of two cuts divided by 100 square microns *Data_type*: float, optional
`-preferred_density percent`	Specifies a percentage value for the preferred cut density. When the cut density in a window is less than the minimum cut density value, this parameter adds via fill to achieve a density as close as possible to the preferred value. To disable the preferred density feature, specify `0`. For example, `set_via_fill -preferred_density 0.02`. *Default:* area of two cuts in the cut layer *Data_type*: float, optional
`-window_size xSize ySize`	Specifies the x and y dimensions in microns of the area `set_via_fill` uses to examine the cut density. The software assesses the cut density in each window, moving the window in iterative manner through the design. For example, `set_via_fill window_size 100 100`. *Default:* 10 microns in each dimension *Data_type*: point, optional
`-window_step xStep yStep`	Specifies the x and y step distance in microns the window moves for each iteration of cut density analysis. For example, `set_via_fill -window_step 50 50`. *Default:* 5 microns in each dimension *Data_type*: point, optional

Example

The following command sets window and density values for inserting via fill in cut layers `VIA1` to `VIA7`:

```
set_via_fill -layer {VIA1 VIA2 VIA3 VIA4 VIA5 VIA6 VIA7} -window_size 100 100 /
          -window_step 50 50 -min_density 0.01 -max_density 30 /
          -preferred_density 1
```

Related Information

The following sections in the Optimizing Metal Density chapter of the *Innovus User Guide* document:

- Overview
- Adding Via Fill

trim_metal_fill

```
trim_metal_fill
[-help]
[-allow_polygon]
[-area x1 y1 x2 y2]
[-delete_violation]
[-ignore_special_net]
[-layer {layer_num_list | layer_name_list}]
[-use_non_default_spacing]
```

Trims metal fill and via fill (FILLWIRE and FILLWIREOPC shapes) that causes DRC violations. This command uses the values set by the set_metal_fill command. Therefore, you must ensure that you have correct set_metal_fill settings before running trim_metal_fill to avoid false violations.

Note: Use set_metal_fill without -iteration_name or with -iteration_name default to set the DRC rules used by trim_metal_fill. set_metal_fill -iteration_name *name*, where *name* is other than default, is used to set construction rules used by add_metal_fill with the -iteration_names option.

Use this command to trim metal fill if the design has floating metal or via fill. By trimming the fill, you reduce the chance of introducing timing problems caused by cross-coupling.

If the design has metal or via fill that is connected to special nets, use fix_open_fill after using trim_metal_fill.

Use the trim_metal_fill command after routing signal nets. For more information, see Route Commands and Attributes.

Parameters

-help	Prints a brief description that includes the type and default information for each trim_metal_fill parameter. For a detailed description of the command and all its parameters, use: man trim_metal_fill
-allow_polygon	Allows rectilinear shapes after trimming metal fill (FILLWIRE and FILLWIREOPC shapes). By default, the Innovus software supports rectangular shapes only. *Data_type*: bool, optional
-area *x1 y1 x2 y2*	Specifies a set of coordinates within which metal fill needs to be trimmed. By limiting the area in which metal fill is trimmed, you can speed up the trimming process. **Note:** This option is recommended for use with only floating metal fill. If you use trim_metal_fill -area on tied-off fill shapes, the command returns a warning message that some of the shapes may become isolated from the Power/Ground network. *Data_type*: rect, optional
-delete_violation	Deletes metal fill shapes that cause DRC violations or shorts, instead of trimming sections of the shapes. After running the trim_metal_fill command with this parameter specified, the remaining shapes are still rectangles. If using this parameter causes metal density to drop below the target density, re-run the add_metal_fill command. Always specify this parameter in the following two cases: • If the design has minimum step rules, specify this parameter to avoid creating minimum step violations. • If you ran add_metal_fill -snap, specify this parameter to delete entire shapes, instead of trimming sections of the shapes, otherwise the trimmed shapes will not be on-grid. *Data_type*: bool, optional
-ignore_special_net	Ignores special nets while checking for DRC violations. *Data_type*: bool, optional
-layer {*layer_num_list* | *layer_name_list*}	Specifies the metal layers in which metal fill needs to be trimmed. **Note:** This option is recommended for use with only floating metal fill. If you use trim_metal_fill -layer on tied-off fill shapes, the command returns a warning message that some of the shapes may become isolated from the Power/Ground network. *Data_type*: string, optional
-use_non_default_spacing	Uses non-default spacing for trimming metal fill. Some critical nets use a non-default rule (NDR) for routing as they may need to have wider wires or wider spacing than other nets. Use -use_non_default_spacing to extend NDR control to trim_metal_fill. *Data_type*: bool, optional

Example

The following command trims metal fill and allows rectilinear shapes:

```
trim_metal_fill -allow_polygon
```

Related Information

- The following sections in the Optimizing Metal Density chapter of the *Innovus User Guide* document:
  - Overview
  - Connected and Floating Metal Fill
  - Trimming Metal Fill
- The 'Shape' section in the DEF Syntax chapter of the *LEF/DEF Language Reference* document

trim_metal_fill_near_net

```
trim_metal_fill_near_net
[-help]
[-area {x1 y1 x2 y2}]
[-clock]
[-create_fill_blockage blockage_name]
[-layers layer_name_list]
[-min_trim_density {percent1 layer_list1 [{percent2 layer_list2}]}]
[-min_trim_union_density_to_above_layer {percent1 layer_list1 [{percent2 layer_list2}]..}]
[-nets {net_name_list | net_object_pointer_list}]
[-recompute_density]
[-remove]
[-slack_threshold slack_in_ns]
[-spacing dist_in_um]
[-spacing_above dist_in_um]
[-spacing_below dist_in_um]
```

Trims any metal fill around critical nets. This command enables you to reduce the timing impact on critical nets due to metal fill on the layer above, below and the same layer for the routing of any critical net. Using this command, you can specify the critical net list, desired spacing around critical nets, and metal fill density limit.

The intended usage model is to run timing analysis before metal fill, add metal fill, then run timing analysis after metal fill, then trim the metal fill that caused timing to get worse. See the examples below.

Parameters

`-help`	Prints a brief description that includes the type and default information for each `trim_metal_fill_near_net` parameter. For a detailed description of the command and all its parameters, use: `man trim_metal_fill_near_net`
`-area {x1 y1 x2 y2}`	Specifies a set of coordinates within which to trim metal fill. If this parameter is not specified, metal fill is trimmed around critical nets in the entire design area. *Data_type*: rect, optional
`-clock`	Specifies whether metal fill is to be trimmed around all clock. *Data_type*: bool, optional
`-create_fill_blockage blockage_name`	Creates a fill blockage so that any subsequent fill steps do not refill the trimmed area. If you specify a name, the tool uses that name as the fill blockage name. If you do not specify a name, the tool uses the default name *netName_layerName_** for the fill blockage. *Data_type*: string, optional
`-layers layer_name_list`	Trims metal fill only on the specified layers. If this parameter is not specified, metal fill is trimmed on all layers. *Data_type*: string, optional
`-nets {net_name_list | net_object_pointer_list}`	Specifies a list of critical net names or pointers which are surrounded by metal fill. *Data_type*: string, optional
`-min_trim_density {percent1layer_list1 [{percent2layer_list2}]}`	Trims metal fill only until the specified window density percentage is reached to avoid causing any metal density violations. The window sizes are controlled by `set_metal_fill. check_metal_density` is used to compute the metal-density using the same set_metal_fill settings. You can specify different window density percentages for different layers. For example, {{20 metal1 metal2} {30 metal3}} would set the min-density for metal1, metal2 windows to 20%, and for metal3 to 30%. *Data_type*: string, optional
`-min_trim_union_density_to_above_layer {percent1 layer_list1 [{percent2 layer_list2}]..}`	Trims metal fill only till meeting the specified percent window density. *Default:* `0` *Data_type*: string, optional

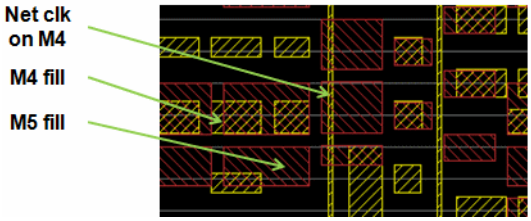
`-recompute_density`	By default, if metal density values are required for `-min_trim_density`, and they are available in the database from a previous call to `check_metal_density`, the existing database values are used. This option forces a call to `check_metal_density` to recompute window densities instead of using the stored values. *Data_type*: bool, optional
`-remove`	Specifies that the metal fill shapes overlapping the critical-nets halo area should be removed entirely instead of being trimmed to the halo edge created with different `-spacing*` options. This avoids the possibility of smaller metal fill shapes that might be left over after trimming. *Data_type*: bool, optional
`-slack_threshold` *slack_in_ns*	Trims metal fill around nets with slack less than the specified value. The value specified should be in nanoseconds. *Data_type*: float, optional
`-spacing` *dist_in_um*	Trims fill on the same layer by specified distance around critical net routes. The distance specified should be in micrometers (`um`). The default is 0 so no trimming is done. *Data_type*: float, optional
`-spacing_above` *dist_in_um*	Trims fill on the layer above the current layer by specified distance around critical net routes. The distance specified should be in micrometers (`um`). The default is 0 so no trimming is done. *Data_type*: float, optional
`-spacing_below` *dist_in_um*	Trims fill on the layer below the current layer by specified distance around critical net routes. The distance specified should be in micrometers (`um`). The default is 0 so no trimming is done. *Data_type*: float, optional

Examples

- The most common usage model is to run timing anlaysis before metal fill, add metal fill, run timing analysis after metal fill, and then trim the metal fill that causes timing to get worse. Suppose the WNS before metal fill was -0.2ns, and after metal fill the WNS is -0.4ns. This means the metal fill caused a 0.2ns degradation. One strategy is to trim any net that got pushed about beyond the original WNS of -0.2ns aggressively with a larger trim spacing value because the extra WNS is directly caused by metal-fill, and then trim any negative slack nets above 0.0ns to improve timing margins. The spacing values depend on the technology node and timing criticality. A reasonable starting point would be to choose between 5x to 10x min-spacing for the aggressive trim value. So you might use these commands:

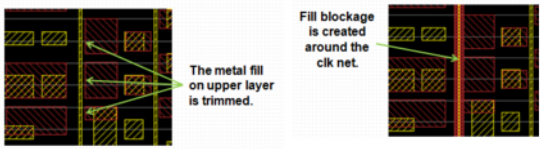
```
#Trim by 1.0um the layer above/below/same for any net with worse slack than the pre-metal fill WNS of -0.2
#Do not trim if the metal density will fall below 20% for metal1, metal2, or 30% for metal3 using the set_metal_fill window settings
trim_metal_fill_near_net -slack_threshold -0.2 -spacing 1.0 -spacing_above 1.0 -spacing_below 1.0 -min_trim_density {{20 metal1 metal2} {30 metal3}}
#Trim less aggressively using 0.4um spacing for any net with WNS above 0.0 to improve timing margins
trim_metal_fill_near_net -slack_threshold 0.0 -spacing 0.4 -spacing_above 0.4 -spacing_below 0.4 -min_trim_density {{20 metal1 metal2} {30 metal3}}
```

- In the following example, there is a critical net overlap with the metal fill on the upper layer, metal5:



The following command trims metal fill on the upper layer, metal5, and creates a fill blockage around the `clk` net.

```
trim_metal_fill_near_net -nets clk -layers {metal4 metal5} -create_fill_blockage -spacing_above 0.2
```



Related Information

The following sections in the Optimizing Metal Density chapter of the *Innovus User Guide* document:

- Overview
- Trimming Metal Fill
- Trimming Metal Fill for Timing Closure

21

Mixed Signal Commands

- delete_integration_route_constraints
- get_integration_route_constraint
- pull_block_constraints
- run_vsr
- set_integration_route_constraint
- write_integration_route_constraints

delete_integration_route_constraints

```
delete_integration_route_constraints
[-help]
[-net net1 net2 ... | -name constraint_name | -custom_constraints  | -all ]
```

Deletes routing constraints in the database.

Note: Integration constraints are used to convey the design intent in a mixed signal design. These constraints are fully interoperable between the digital and custom environments.

Parameters

`-help`	Prints a brief description that includes the type and default information for each `delete_integration_route_constraints` parameter. For a detailed description of the command and all of its parameters, use the man command `man delete_integration_route_constraints`.
`-all`	Deletes all integration constraints, including constraints that are understood by NanoRoute and used to route nets with simple shield or NDR constraints. *Data_type*: bool, optional
`-custom_constraints`	
	Deletes all diffPair, matchLength, complex shield, and complex netClass constraints. *Data_type*: bool, optional
`-name` *constraint_name*	Allows the deletion of a specific constraint by name. You can specify multiple constraint names, enclosed in braces. Use the `get_integration_route_constraint` command to get the name of any constraint in the design. *Data_type*: string, required
`-net` *net1 net2 ...*	Indicates the nets for which the constraints will be deleted. If the net is part of a netClass, the netClass will be deleted as well. *Data_type*: string, required

Examples

- The following command deletes all diffPair, matchLength, complex shield, and complex netClass constraints.

  ```
  delete_integration_route_constraints -custom_constraints
  ```

- The following command deletes the constraints with the specified names:

  ```
  delete_integration_route_constraints -name {Constr_2__analogNW__DTMF_INST/analogNW_INST
  Constr_1__analogNW__DTMF_INST/analogNW_INST}
  ```

- The following command deletes all Integration constraints in the design.

  ```
  delete_integration_route_constraints -all
  ```

  The tool displays the following warning before deleting the constraints:

  ```
  **WARN: (ENCMSC-270): You are attempting to delete all routing constraints in this design.
  These include many constraints that are intended for use by NR for routing nets with simple
  shield, or NDR constraints.
  ```

Related Information

- `get_integration_route_constraint`
- `set_integration_route_constraint`
- `write_integration_route_constraints`
- Routing Constraint Interoperability chapter in the *Mixed Signal Interoperability Guide*
- Tools Menu chapter in the *Menu Reference*
  - Mixed Signal -> Integration Constraint Editor

get_integration_route_constraint

```
get_integration_route_constraint
[-help]
[-net net_name | -name constraint_name]
```

Displays settings needed for the `set_integration_route_constraint` command.

If mixed-signal constraints have not yet been defined, the `get_integration_route_constraint` command does not return anything.

If you run the `get_integration_route_constraint` command without any option, all nets with all constraint names are displayed on the console window.

Note: The output is displayed only on the console window and is not written to the log file. You cannot use `>` to write to a file. The only way to send the output to a file is to run the following:

```
redirect {puts [get_integration_route_constraint xxxx]} > filename
```

Alternatively, use the command `write_integration_route_constraints`.

Note: Integration constraints are used to convey the design intent in a mixed-signal design. These constraints are fully interoperable between the digital and custom environments.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `get_integration_route_constraint` parameter. For a detailed description of the command and all of its parameters, use the man command `man get_integration_route_constraint`.
`-name constraint_name`	Returns a specific constraint by name. You can specify multiple constraint names, enclosed in braces. *Data_type*: string, optional
`-net net_name`	Specifies the net for which the constraints will be returned. If the net is part of a net_class, the constraints for the net_class is returned. You can specify multiple net names, enclosed in braces. *Data_type*: string, optional

Example

- The following command returns the constraint(s) defined for the net `scan_out_1`:

```
get_integration_route_constraint -net scan_out_1
```

The output of the command is:

```
{-type shield  -shield_net { VSS } -shield_type side -shield_tolerance 30.0 -net {
scan_out_1 }
}
```

- The following command returns details of the constraint named `Constr_13__analogSW__DTMF_INST/analogSW_INST`:

```
get_integration_route_constraint -name Constr_13__analogSW__DTMF_INST/analogSW_INST
```

The output of the command is:

```
{-type net_class -name Constr_13__analogSW__DTMF_INST/analogSW_INST -shield_net VSS -
shield_type side -connect_supply anypoint -rule NDR8 -net {analogW4I}
}
```

- The following command returns the constraints on the specified nets:

```
get_integration_route_constraint -net {DTMF_INST/cntrlxo[3] DTMF_INST/cntrlxo[0]}
```

The output of the command is:

```
{-type diff_pair -name Constr_2__analogNW__DTMF_INST/analogNW_INST -tolerance 20 -rule
default -net {DTMF_INST/cntrlxo[3] DTMF_INST/cntrlxo[2]}
} {-type diff_pair -name Constr_1__analogNW__DTMF_INST/analogNW_INST -tolerance 5 -rule
default -net {DTMF_INST/cntrlxo[0] DTMF_INST/cntrlxo[1]}
}
```

- The following command returns all the constraints on all the nets in the design:

```
get_integration_route_constraint
```

A sample output of the command is displayed below:

```
{-type diff_pair -name Constr_2__analogNW__DTMF_INST/analogNW_INST -tolerance 20 -rule
default -net {DTMF_INST/cntrlxo[3] DTMF_INST/cntrlxo[2]}
} {-type diff_pair -name Constr_1__analogNW__DTMF_INST/analogNW_INST -tolerance 5 -rule
default -net {DTMF_INST/cntrlxo[0] DTMF_INST/cntrlxo[1]}
} {-type diff_pair -name Constr_2__analogNL__DTMF_INST/analogNL_INST -tolerance 20 -rule
default -net {DTMF_INST/digsynca DTMF_INST/digsyncb}
} {-type diff_pair -name Constr_3__analogNL__DTMF_INST/analogNL_INST -tolerance 20 -rule
default -net {DTMF_INST/digsyncc DTMF_INST/digsyncd}
} {-type match_length -name Constr_5__analogNL__DTMF_INST/analogNL_INST -tolerance 20 -
match_style accordion -shield_net VSS -shield_type unspecified -connect_supply anypoint -
shield_width { Metal1 0.56 Metal2 0.56 Metal3 0.56 Metal4 0.56 Metal5 0.88 Metal6 0.88} -
rule default -net {DTMF_INST/digsync2 DTMF_INST/digsync1} hasUnknownConstraint
} {-type match_length -name Constr_4__analogNL__DTMF_INST/analogNL_INST -tolerance 20 -
match_style accordion -rule default -net {DTMF_INST/digsync3 DTMF_INST/digsync4}
} {-type net_class -name Constr_13__analogSW__DTMF_INST/analogSW_INST -shield_net VSS -
shield_type side -connect_supply anypoint -rule NDR8 -net {analogW4I}
} {-type net_class -name Constr_12__analogSW__DTMF_INST/analogSW_INST -shield_net VSS -
shield_type side -connect_supply anypoint -rule NDR8 -net {analogW3I}
```

```
} {-type net_class -name Constr_11__analogSW__DTMF_INST/analogSW_INST -shield_net VSS -
shield_type side -connect_supply anypoint -rule NDR8 -net {analogW2I}
} {-type net_class -name Constr_10__analogSW__DTMF_INST/analogSW_INST -shield_net VSS -
shield_type side -connect_supply anypoint -rule NDR8 -net {analogW1I} hasUnknownConstraint
} {-type shield -shield_net { VSS } -shield_type side -net { analogS4I }
hasUnknownConstraint
} {-type shield -shield_net { VSS } -shield_type side -net { analogS3I }
hasUnknownConstraint
} {-type shield -shield_net { VSS } -shield_type unspecified -net { analogS2I }
} {-type shield -shield_net { VSS } -shield_type side -net { analogS1I }
} {-type shield -rule NDR0.6__analogNE -shield_net { AVSS } -shield_type coaxial -
connect_supply anypoint -shield_width { Metal1 0.23 Metal2 0.28 Metal3 0.28 Metal4 0.28
Metal5 0.28 Metal6 0.44} -tandem_width { Metal1 1.4 Metal2 1.4 Metal3 1.4 Metal4 1.4 Metal5
1.4 Metal6 1.4} -shield_gap { Metal1 0.23 Metal2 0.28 Metal3 0.28 Metal4 0.28 Metal5 0.28
Metal6 0.46} -net { analogE1I } hasUnknownConstraint
} {-type shield -shield_net { AVSS } -shield_type coaxial -connect_supply anypoint -
shield_width { Metal1 0.23 Metal2 0.28 Metal3 0.28 Metal4 0.28 Metal5 0.28 Metal6 0.44} -
tandem_width { Metal1 1.4 Metal2 1.4 Metal3 1.4 Metal4 1.4 Metal5 1.4 Metal6 1.4} -
shield_gap { Metal1 0.23 Metal2 0.28 Metal3 0.28 Metal4 0.28 Metal5 0.28 Metal6 0.46} -net
{ analogN4I } hasUnknownConstraint
} {-type shield -shield_net { AVSS } -shield_type coaxial -connect_supply anypoint -
shield_width { Metal1 0.23 Metal2 0.28 Metal3 0.28 Metal4 0.28 Metal5 0.28 Metal6 0.44} -
tandem_width { Metal2 1.4 Metal3 1.4 Metal4 1.4 Metal5 2.24 Metal6 1.4} -shield_gap {
Metal1 0.23 Metal2 0.28 Metal3 0.28 Metal4 0.28 Metal5 0.28 Metal6 0.46} -net { analogN3I }
hasUnknownConstraint
} {-type shield -shield_net { AVSS } -shield_type coaxial -connect_supply anypoint -
shield_width { Metal1 0.23 Metal2 0.28 Metal3 0.28 Metal4 0.28 Metal5 0.28 Metal6 0.44} -
tandem_width { Metal1 1.4 Metal2 1.4 Metal3 1.4 Metal4 1.4 Metal5 1.4 Metal6 1.4} -
shield_gap { Metal1 0.23 Metal2 0.28 Metal3 0.28 Metal4 0.28 Metal5 0.28 Metal6 0.46} -net
{ analogN2I } hasUnknownConstraint
} {-type shield -rule NDR0.6__analogNE -shield_net { AVSS } -shield_type coaxial -
connect_supply anypoint -shield_width { Metal1 0.23 Metal2 0.28 Metal3 0.28 Metal4 0.28
Metal5 0.28 Metal6 0.44} -tandem_width { Metal1 1.4 Metal2 1.4 Metal3 1.4 Metal4 1.4 Metal5
1.4 Metal6 1.4} -shield_gap { Metal1 0.23 Metal2 0.28 Metal3 0.28 Metal4 0.28 Metal5 0.28
Metal6 0.46} -net { analogN1I } hasUnknownConstraint
}
```

Related Information

- `delete_integration_route_constraints`
- `set_integration_route_constraint`
- `write_integration_route_constraints`
- Routing Constraint Interoperability chapter in the *Mixed Signal Interoperability Guide*
- Tools Menu chapter in the *Menu Reference*
  - Mixed Signal -> Integration Constraint Editor

pull_block_constraints

```
pull_block_constraints
[-help]
[-override]
[-report_file report_file_name]
{-all_blocks | -block list_of_blocks | -inst list_of_instances | -cell_view
{list_of_libName_cellName_viewName}}
```

Pulls the routing constraints stored on the interface nets of blocks in a design, to their corresponding top-level nets. The `pull_block_constraints` command uses a two-pass approach to handle conflicting constraints on the same net. In this approach, the `pull_block_constraints` command detects block-level and top-level conflicts and reports a warning.

If the `-override` parameter is not specified (non-override mode) and a block-level conflict is detected, no constraints are pulled.

If the `-override` parameter is specified (override mode), `pull_block_constraints` reports the conflicts but pulls the constraints such that the lower-level net constraints override any constraints that might exist on the top-level nets..

Parameters

`-help`	Prints a brief description that includes the type and default information for each `pull_block_constraints` parameter. For a detailed description of the command and all of its parameters, use the man command `man pull_block_constraints`
`-all_blocks`	Specifies that constraints with hierarchical scope will be pulled from all blocks in the design. *Data_type*: bool, required
`-block` *list_of_blocks*	Defines the list of block names of the instances in the design for which the constraints with hierarchical scope will be pulled. *Data_type*: string, required
`-cell_view {`*list_of_libName_cellName_viewName*`}`	
	Specifies the list of library, cell, and view names from which the constraints need to be pulled. *Data_type*: string, required

`-inst` *`list_of_instances`*	Defines the instances in the design for which the constraints with hierarchical scope will be pulled. *Data_type*: string, required
`-override`	When specified, the lower-level net constraints will override any constraints that might exist on the top-level nets. *Default*: `off` *Data_type*: bool, optional
`-report_file` *`report_file_name`*	Specifies the report file name. Given that the standard out report is in the abbreviated format, more details are written out in this mode indicating: • The lower, and upper level net names, the type of constraints pulled. • All the associated parameters. • Any conflicts that need to be resolved. • Which constraint were written to a particular net. *Default*: `dumpPullInfo.log` *Data_type*: string, optional

Examples

- Using the following command, the constraints on the pins in the cellview binded to the block macro of all the instantiations of the block list `pll_clock` and `pwr_monitor` will be pulled up to the top-level nets connected to the instance pins.
  ```
  pull_block_constraints -block {pll_clock pwr_monitor}
  ```
- Using the following command, the constraints on the pins in the cellview binded to the macro instance `I1/I2` will be pulled up to the nets connected to the pins of the instance.
  ```
  pull_block_constraints -inst {I1/I2}
  ```

Related Information

- `delete_integration_route_constraints`
- `write_integration_route_constraints`
- Routing Constraint Interoperability chapter in the *Mixed Signal Interoperability Guide*
- Tools Menu chapter in the *Menu Reference*

- Mixed Signal -> Pull Block Constraint

run_vsr

```
run_vsr
[-help]
[-log_file fileName]
[-no_taper_to_pin_width]
[-post_script scriptname]
[-pre_script scriptname]
[-remove_existing_routes]
[-route_type_order {bus diffpair match nets shield}]
[-run_vsr_script scriptname]
[-share_shields]
[-nets list_of_selected_nets | -include_net_types {bus diffpair match nets shield} | -nets_file
fileName]
```

Launches the Virtuoso Space-based Router (VSR) from Innovus. It works on OpenAccess design as well as restored non-OpenAccess designs. It routes the nets specified with the `-nets`, `-nets_file`, or `-include_net_types` option. If the `-include_net_types` option is used, the nets are routed in the following order: `diffpair`, `match`, `bus`, `nets`, and `shield`. To invoke this wrapper, make sure Virtuoso is available in your path.

Before running the `run_vsr` command, set the following:

```
set_db oa_new_lib_compress_level 1
```

Notes

- Integration constraints are used to convey the design intent in a mixed-signal design. These constraints are fully interoperable between the digital and custom environments.
- If you run `run_vsr` on an abstract inferred from layout, you may see the following WARNING message:

  ```
  "The current database uses quick abstract inference and should be used only for
  floorplanning and not for routing purposes."
  ```

For more information on using `run_vsr`, see the Routing Constraint Interoperability chapter in the *Mixed Signal Interoperability Guide*.

Parameters

`-help`	Prints a brief description that includes the type and default information for each `run_vsr` parameter. For a detailed description of the command and all of its parameters, use the man command `man run_vsr`
`-include_net_types {bus diffpair match nets shield}`	

	Specifies the list of valid net types for VSR to route. **Note:** The `-nets`, `-nets_file`, and `-include_net_types` parameters are mutually exclusive. Do not specify these parameters at the same time. *Data_type*: (enum)+, optional
`-log_file` *fileName*	Specifies the file name for writing the routing report. *Data_type*: string, optional
`-nets` *list_of_selected_nets*	Specifies the list of nets that have to be routed. *Data_type*: string, optional
`-nets_file` *fileName*	Specifies the name of the file listing the nets to be routed. **Note:** The `-nets`, `-nets_file`, and `-include_net_types` parameters are mutually exclusive. Do not specify these parameters at the same time. *Data_type*: string, optional
`-no_taper_to_pin_width`	Does not perform tapering based on the pin width. For instance, if the rule applied on a mixed signal net is NDR1 whose width exceeds the pin width for a particular layer, then VSR automatically tapers the wire to match the pin width. If this option is specified, then tapering is not done. *Data_type*: bool, optional
`-post_script` *scriptname*	Specifies the name of the script to be executed by VSR after loading the design. *Data_type*: string, optional
`-pre_script` *scriptname*	Specifies the name of the script to be executed by VSR before loading the design. *Data_type*: string, optional
`-remove_existing_routes`	Deletes the nets (fixed, normal), including shield, and excluding power nets, which are specified by the -nets parameter before routing. *Data_type*: bool, optional
`-route_type_order {bus diffpair match nets shield}`	
	Specifies the list of valid constraint types in the preferred routing order. *Data_type*: (enum)+, optional

`-run_vsr_script` *scriptname*	Specifies a user-defined script to be run by VSR for custom routing. If this option is specified, the router skips all presetting from VSR and routes the design by sourcing the complete custom script as provided. *Data_type*: string, optional
`-share_shields`	Shares the shielded nets. Using this parameter, you can selectively specify the list of shared shielded nets. *Data_type*: bool, optional

Using Scripts and Environmental Variables with run_vsr

The `run_vsr` command in Innovus supports a limited set of options to route the analog/mixed-signal nets. To have more control of the VSR engine, you can set additional settings through the `.cdsenv` file.

Settings through .cdsenv

Here are two examples of how you can use `.cdsenv` settings to control VSR:

- To force VSR to route on-grid:
  ```
  layout routeUseGrid cyclic "Routing"
  ```
- To force VSR to use the rules from the specified route spec:
  ```
  layout wireConstraintGroup string "LEFDefaultRouteSpec"
  ```

If you set the above settings in the `.cdsenv` file before running the `run_vsr` command, Virtuoso initializes them before calling the VSR engine.

In addition to the settings through `.cdsenv` file, the `run_vsr` command is enhanced to pass additional variables or scripts through `-pre_script` and `-post_script`.

Preloadscript

Usage of settings through `-pre_script` is only for expert VSR users and generally not recommended.

Postloadscript

Example:

- To enable VSR to use the custom vias and not generated or derived vias:
  ```
  setvar db.enable_create_derived_vias false
  ```
- To enable VSR to connect the wires to the center of the pin
  ```
  setvar droute.snap_to_pin_center true
  ```

You can set the above environmental variables in a script and enable them through the following command:

```
run_vsr -nets { neta …..} -post_script scriptname…
```

For more information on the environmental variables which can be set, refer to the following documentation:

- "Environment Variables" chapter in the *Cadence Space-based Router and Chip Optimizer Command Reference*

Note: Consult Cadence for the best use of these variables to obtain the desired results.

Examples

- The following command routes the bus nets using VSR:
  ```
  run_vsr -nets {DATA[0] DATA[1] DATA[2] DATA[3] DATA[4] DATA[5] DATA[6] DATA[7]}
  ```
- The following command routes the given net `lbk_bstart` using mixed-signal constraint, if any constraint has been applied to this net which will be taken from OpenAccess database:
  ```
  run_vsr -nets lbk_bstart
  ```
- The following command routes all the nets which have `diffpair`, `matchpair` and then `net` class in the following order of constraint types:
  ```
  run_vsr -include_net_types {diffpair match nets}
  ```

Related Information

- Routing Constraint Interoperability chapter in the *Mixed Signal Interoperability Guide*
- Tools Menu chapter in the *Menu Reference*
  - Mixed Signal -> Run VSR

set_integration_route_constraint

```
set_integration_route_constraint
[-help]
[-access_style {none straight clustered}]
[-bottom_layer layerNum]
[-bus_group_width integer]
[-connect_supply {float anypoint}]
[-corner_style {none crossover river}]
[-crosstie_shield_interval string]
[-crosstie_shield_layer_limit string]
[-crosstie_shielded_layers string]
[-group_to_outside_spacing {layer1 spacing1 layer2[:layerN] spacing2...}]
[-hard]
[-hierarchical_scope {local local_above local_below local_above_below}]
[-inter_group_gap {layer1 spacing1 layer2[:layerN] spacing2...}]
[-layer_gap {layer1 gap1 layer2[:layerN] gap2...}]
[-layer_match]
[-match_gap {layer1 gap1 layer2[:layerN] gap2...}]
[-match_style {accordion | trombone | coil}]
[-max_length value]
[-max_res value]
[-min_bus_bit integer]
[-min_length value]
[-min_res value]
[-name string]
{-net {net1 net2 ...} | -bus_sink_groups bus_sink_group_name_list}
[-no_share_shield]
[-rule NDR_name]
[-shield_gap {layer1 gap1 layer2[:layerN] gap2 ...}]
[-shield_net {net1 net2}]
[-shield_tolerance {net1 value1 net2 value2 ...}]
[-shield_type {unspecified side bottom bottom_side top top_side top_bottom coaxial tandem_split
               interleaved crosstie tandem_split_full tandem_split_nearby tandem_split_special}]
[-shield_width {layer1 width1 layer2[:layerN] width2...}]
[-tandem_aggressor_adjacent_layer_separation layer separation]
[-tandem_aggressor_same_layer_track_distance {layer1 n1 layer2[:layerN] n2...}]
[-tandem_cap_width {layer1 width1 layer2[:layerN] width2...}]
[-tandem_nearby_non_aggressor_except_by_name {nameString}]
[-tandem_nearby_non_aggressor_nets {nameString}]
[-tandem_nearby_non_aggressor_type {typeNameString}]
[-tandem_overhang {layer1 width1 layer2[:layerN] width2...}]
[-tandem_side_overhang {layer1 width1 layer2[:layerN] width2...}]
[-tandem_width {layer1 width1 layer2[:layerN] width2...}]
[-tolerance tolerance]
[-top_layer layerNum]
```

`-type {bus diff_pair match_length ndr net_class shield simple_net}`

Applies specialty routing constraints in the database for nets in the designs. It allows creation of differential pair, match pair, shield nets, and nets with non-default rules.

If required, all routing constraints set by this command can be deleted from the database using the `delete_integration_route_constraints` command.

Also see, `get_integration_route_constraint`.

Notes

- Integration constraints are used to convey the design intent in a mixed signal design. These constraints are fully interoperable between the digital and custom environments.
- If the design has a complex `net_class` constraint that has net members with different constraints, `set_integration_route_constraint` will throw the an error. If you still wish to change the integration constraints in the design, check whether the `net_class` constraints under the *Complex NetClass* tab in the Integration Constraint Editor can be removed. If yes, use the `delete_integration_route_constraints` or the GUI to first delete such constraints, and then repeat the `set_integration_route_constraint` command.
- The rule name `DefaultRule` is obsolete. It still works in this release and is automatically converted to the correct name `default`. Update your scripts to use `default` so that it work correctly in future releases.

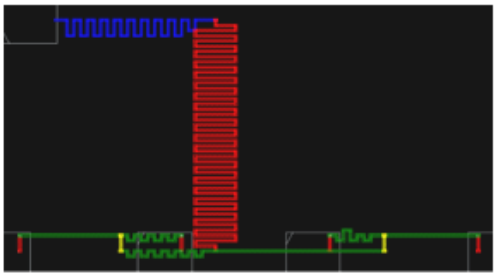
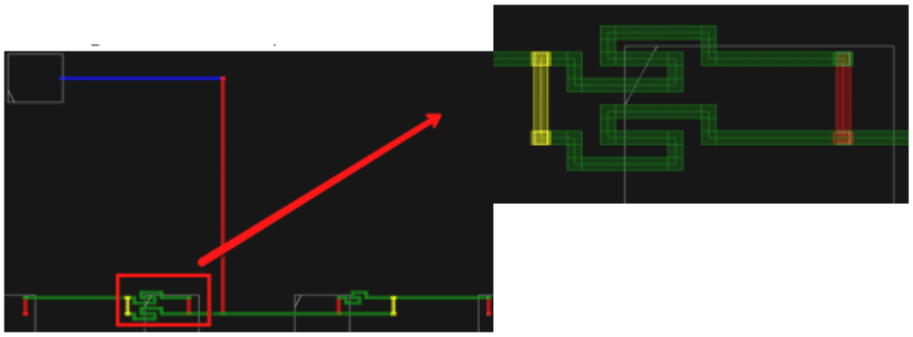
Parameters

`-help`	Outputs a brief description that includes the type and default information for each `set_integration_route_constraint` parameter. For a detailed description of the command and all of its parameters, use the man command `man set_integration_route_constraint`.
`-access_style {none straight clustered}`	
	Specifies the preferred style for routing the bus after connecting to the bus pins. This parameter is applicable only for`-type bus`. *Default:* `clustered` *Data_type*: enum, optional
`-bottom_layer` *layerNum*	Specifies the bottom routing layer that is to be used. *Data_type*: string, optional
`-bus_group_width` *integer*	

	Specifies the width for the bus groups into which the router splits a bus. The width is specified as an integer value. For example, if a bus is 16 bits, you can specify `-bus_group_width 4` to indicate that the router can route the bus as four bus groups of 4 bits each. *Default*: `1` *Data_type*: int, optional
`-bus_sink_groups` *bus_sink_group_name_list*	
	Specifies the list of names of the busSinkGroups. It sets high-frequency/mixed signal constraints for the nets of the busSinkGroup sinks. It supports all constraints of type BUS. These constraints are honored by the NanoRoute high-frequency router. *Data_type*: string, optional
`-connect_supply {`*float* `anypoint}`	
	Indicates how the shield wires will be connected to the supply line. The default option, `anypoint`, tells the router to connect the wire to supply, anywhere along the length of the shield wire. *Default:* `anypoint` *Data_type*: enum, optional
`-corner_style {none crossover river}`	
	Specifies the type of corner style that the differential pair route or bus should follow when changing directions. This parameter is applicable only for `-type bus` and `-type diff_pair`. *Default:* `crossover` *Data_type*: enum, optional
`-crosstie_shield_interval` *string*	

	Specifies the number of tracks between the cross-tie wires for opportunistic coaxial shielding. The *string* value should have the following format: `{integer | {layer interval1 layer2[:layerN] interval2 ...}}` The integer value specifies the interval in the form of the number of tracks between cross-ties created on a given layer. For example, a value of `0` means that there are no tracks between cross-ties, which means there is a cross-tie on every track. Similarly, a value of 1 means that there is one track between cross-tie wires, which results in a cross tie on every other track. A value of `2` means two tracks between the cross-tie wires and so on. **Note**: This parameter is applicable only if `-shieldType` is specified as `crosstie`. *Default:* `0` *Data_type*: string, optional
`-crosstie_shield_layer_limit` *string*	
	Specifies the layer range that can be used to create cross-tie shields. Use the `-crosstie_shield_layer_limit` parameter if you want to use only specific layers to create cross-ties and keep the remaining layers free for other purposes. The *string* value should have the following format: `{layer_number:layer_number}` For example, if the layer range specified is `{METAL2:METAL7}`, cross-tie shields can be created on `METAL2`, `METAL3`, `METAL4`, `METAL5`, `METAL6`, and `METAL7` but not on `METAL8` or the layers above it. **Note**: This parameter is applicable only if `-shieldType` is specified as `crosstie`. *Data_type*: string, optional
`-crosstie_shielded_layers` *string*	
	Specifies the layer range that will be shield with cross-ties from above and below. Use the `-crosstie_shielded_layers` parameter if you want to minimize congestion by shielding only specific routing layers with cross-ties. The *string* value should have the following format: `{layer:layer}` For example, if the layer range specified is `{METAL3:METAL6}`, cross-tie shields will be built for routes on the net on `METAL3`, `METAL4`, `METAL5`, and `METAL6`. **Note**: This parameter is applicable only if `-shieldType` is specified as `crosstie`. *Data_type*: string, optional

`-group_to_outside_spacing` *layer1 spacing1 layer2*`[:`*layerN*`]` *spacing2*`...`	
	Indicates the spacing per layer (in microns) between the nets in this net group, and the other nets in the design. *Data_type*: string, optional
`-hard`	Indicates that the top or bottom layer routing constraint created should be of hard type. By default, Innovus places the top/bottom routing layer constraints as soft routing constraints in the resulting OpenAccess database. *Data_type*: bool, optional
`-hierarchical_scope {local local_above local_below local_above_below}`	
	Specifies whether the particular constraint is local, and/or can be pulled (above) or pushed (below) across the hierarchy. Setting this option to all possible values enables pull and push. If set to local, it is not saved to the database. *Data_type*: enum, optional
`-inter_group_gap {`*layer1 spacing1 layer2*`[:`*layerN*`]` *spacing2*`...}`	
	Specifies the gap to be kept between the split bus groups when `-bus_group_width` is specified. Default is min spacing. *Data_type*: string, optional
`-layer_gap {`*layer1 gap1 layer2*`[:`*layerN*`]` *gap2*`...}`	
	Indicates the layer names and the gap-spacing values for the nets routed as `diff_pair`. For any layer not listed, the minimum legal spacing for that layer is used. For example: `-layer_gap {`*layer_1 gap_1 layer_2 gap_2*`...}` *Data_type*: string, optional
`-layer_match`	Specifies that each separate layer should be matched for `match_length` nets. By default, only the total length is matched. *Data_type*: bool, optional
`-match_gap {`*layer1 gap1 layer2*`[:`*layerN*`]` *gap2*`...}`	
	Indicates the spacing between two rises of the accordion. This setting is applicable only when `-type` is `match_length` and `-match_style` is `accordion`. You may want to control the spacing between two rises of the accordion if routing resources are restricted. *Data_type*: string, optional

`-match_style {accordion | trombone | coil}`	
	Indicates the routing topology to be used for matching the nets routed as `match_length`. It can be `accordion`, `trombone`, or `coil`. In `accordion`, pulse structures are used to match length while in `trombone`, loops are used. **Matching in accordion style**  **Matching in trombone style**  *Default:* `accordion` *Data_type*: enum, optional
`-max_length` *value*	Indicates the maximum length the router should route a net. The maximum length is specified as a float value. This parameter is applicable only for `-type match_length` and `-type simple_net`. *Data_type*: float, optional
`-max_res` *value*	Indicates the maximum resistance to be used by the Virtuoso Space-based Router (VSR) for the specific net being routed, or for the nets with a match_length constraint if `-type` is `match_length`. The maximum resistance is specified as a float value. This parameter is applicable only for `-type match_length`. *Data_type*: float, optional

`-min_bus_bit` *integer*	
	Indicates the minimum bits allowed for the router to split a bus. As channels get narrower, it becomes challenging to accommodate all the bus bits in the same channel. By default, `-min_bus_bit` value is undefined, not `0`. If you specify an integer value for `-min_bus_bit`, the router splits a bundle with a minimum of this value. For example, if you set `-min_bus_bit` to `4`, the router splits the bus such that each part has at least `4` bits. *Data_type*: int, optional
`-min_length` *value*	Indicates the the minimum length the router should route a net. The minimum length is specified as a float value. This parameter is applicable only for `-type match_length` and `-type simple_net`. *Data_type*: float, optional
`-min_res` *value*	Indicates the minimum resistance to be used by VSR for the specific net being routed, or for the nets with a `match_length` constraint if `-type` is `match_length`. The minimum resistance is specified as a float value. This parameter is applicable only for `-type match_length`. *Data_type*: float, optional
`-name` *string*	Specifies the name of the constraint group being created, changing the default name given to the constraint by Innovus. *Data_type*: string, optional
`-net {`*net1 net2*`}`	Specifies the list of net names to be routed. *Data_type*: string, optional
`-no_share_shield`	Specifies that there should be individual shields, instead of a common shield, between the bus nets. This option is applicable only for constraints of type `bus`.
`-rule` *NDR_name*	Specifies the name of the routing rule that is used for routing a particular type of net(s). *Default:* `"default"` *Data_type*: string, optional
`-shield_gap {`*layer1 gap1 layer2*`[:`*layerN*`] gap2 ...}`	
	Specifies the spacing between the shielded wire and the shield net. For any layer not listed, the minimum legal spacing for that layer is used. *Data_type*: string, optional

`-shield_net {net1 net2}`	Indicates that the net group should be shielded by the specified nets. Valid net names that you can use with `-shield_net` are all power and ground net names in the design. You can specify different shield nets for different sides of a wire. The routers then determine which shield net should be used on which side of the wire. **Note:** You can specify a maximum of two net names with this parameter. In addition, you can specify two shield net names only for the following shield types: • `side` • `top_side` • `bottom_side` • `coaxial` • `tandem_split` For the `top`, `bottom`, or `top_bottom` shield types, you can specify only one shield net. *Default*: The first ground net name in the list of ground nets. *Data_type*: string, optional
`-shield_tolerance {net1 value1 net2 value2 ...}`	

Indicates the percentage of length of the net that can remain unshielded. You can set the same tolerance value or different tolerance values on a group of nets.

The examples below depict different use models:

- In the following example, the same constraint shield tolerance is put on every member of the shield constraint group:

```
set_integration_route_constraint -type shield -shield_net {VSS} -
shield_type side -shield_tolerance 25.0 -connect_supply anypoint -net
{scan_in_2, abc, xyz}
```

- In the example below, different groups of nets get different shield_tolerance values:

```
set_integration_route_constraint -type shield -shield_net {VSS} -
shield_type side -shield_tolerance 25.0 -connect_supply anypoint -net
{scan_in_2, abc, xyz} -shield_tolerance {net1 30 net2 30 net3 30}
```

- In the following example, the nets `scan_in_2`, `abc` and `xyz` are already nets with the shield constraint, but without tolerance values. The command below adds the tolerance value to the existing shielding constraint of the nets. Note that adding shield tolerance changes the net to a complex shield as opposed to a simple shield:

```
set_integration_route_constraint -type shield -shield_tolerance 25 -net
{scan_in_2 abc xyz}
```

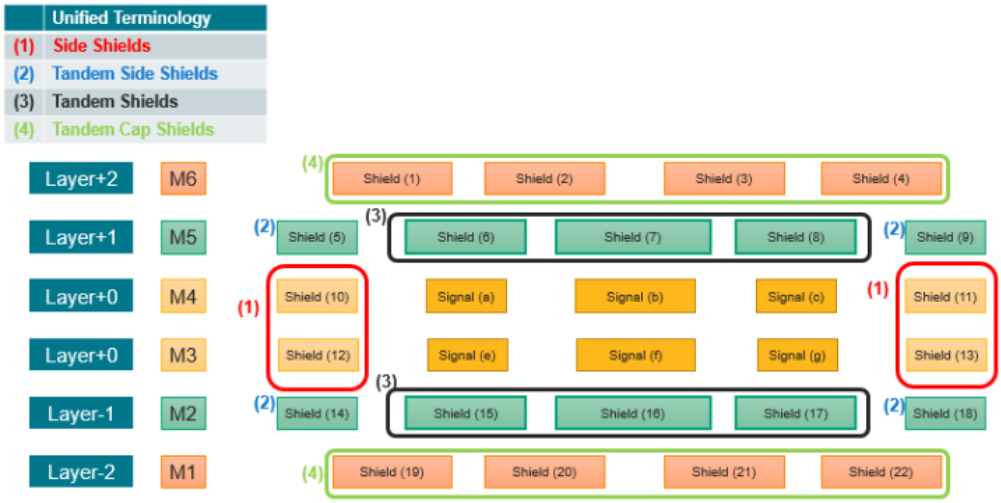
Note:

- The shield tolerance value can be set only on nets that are not intended to be routed by NanoRoute. This means that setting the shield tolerance on any net will make that net a complex shield.
- If you use the `set_integration_route_constraint` command to set the shield tolerance on a net or a group of nets, those nets automatically get the MS bit in the DB and show up in the GUI as a complex shield.
- Percentage tolerance is considered by VSR as a soft constraint.
- PVS-cv is the only tool in the flow that checks the percentage tolerance against the geometries created by VSR, and flags any such violations.

Default: `20.0`

Data_type: string, optional

`-shield_type {unspecified side bottom bottom_side top top_side top_bottom coaxial tandem_split interleaved crosstie tandem_split_full tandem_split_nearby tandem_split_special}`

	Specifies the shielding type for the net(s) being routed. For complex coaxial shields, shield type can be specified as explained below: • `tandem_split_nearby` - Specifies that shielding should be added only where required. • `tandem_split_full` - Specifies that one-layer tandem shielding with overhang should be added. The overhang can be specified using the new `-tandem_overhang` and `-tandem_side_overhang` parameters. • `tandem_split_special` - Specifies that two-layer tandem shielding should be added for complete shielding from other nets. Use the new `-tandem_cap_width` parameter to specify the width of the tandem cap shield (second layer shield).  *Default:* `unspecified` for `diff_pair`, `match_length`, and `bus`; `side` for `NDR`, `simple_net`, and `net_class` *Data_type*: enum, optional
`-shield_width {`*`layer1 width1 layer2`*`[:`*`layerN`*`] `*`width2...`*`}`	
	Indicates the width of the shield wire in microns per layer. This only applies to side shields. *Data_type*: string, optional
`-tandem_aggressor_adjacent_layer_separation` *`layer separation`*	

	Specifies the minimum distance between the given signal layer and the above/below layers containing the aggressors. *Data_type*: string, optional
`-tandem_aggressor_same_layer_track_distance {`*`layer1 n1 layer2`*`[:`*`layerN`*`] `*`n2`*`...}`	
	Specifies the minimum pitch tracks (distance) between the given signal net and any aggressors on the same layer. *Data_type*: string, optional
`-tandem_cap_width {`*`layer1 width1 layer2`*`[:`*`layerN`*`] `*`width2`*`...}`	
	Specifies the wire width (in microns) of the tandem cap shield beyond the edge of the route being shielded. For any layer not listed, the minimum legal width for that layer is used. This parameter is needed only for the `tandem_split_special` shield type. *Data_type*: string, optional
`-tandem_nearby_non_aggressor_except_by_name {`*`nameString`*`}`	
	Specifies that only the nets mentioned by name are aggressors; the rest are not considered as aggressors for shielding. *Data_type*: string, optional
`-tandem_nearby_non_aggressor_nets {`*`nameString`*`}`	
	Specifies the net names that are not considered as aggressors for shielding. *Data_type*: string, optional
`-tandem_nearby_non_aggressor_type {`*`typeNameString`*`}`	
	Specifies those signal types that are not considered as signal types of aggressors. Even if they exist within a certain range, the signal net does not need to be protected by shielding. *Data_type*: string, optional
`-tandem_overhang {`*`layer1 width1 layer2`*`[:`*`layerN`*`] `*`width2`*`...}`	
	Specifies the overhang value (in microns) for the tandem shield. *Data_type*: string, optional
`-tandem_side_overhang {`*`layer1 width1 layer2`*`[:`*`layerN`*`] `*`width2`*`...}`	
	Specifies the overhang value (in microns) for the tandem side shield. *Data_type*: string, optional

`-tandem_width {`*`layer1 width1 layer2`*`[:`*`layerN`*`] `*`width2...`*`}`	
	Specifies the overhang value for the shields on the top and bottom layers so as to extend them to overlap the side shields. *Data_type*: string, optional
`-tolerance` *tolerance*	Indicates the percentage difference in length that is allowed when routing nets as `match_length` and `diff_pair`. *Default:* 20.0, *Min*: 0, *Max*: 100 *Data_type*: float, optional
`-top_layer` *layerNum*	Specifies the top routing layer that is to be used for routing. *Data_type*: string, optional
`-type {bus diff_pair match_length ndr net_class shield simple_net}`	
	Specifies the type of routing constraints to be created: • `bus` - Specifies constraints for bus nets. • `diff_pair` - Specifies constraints for differential pair nets. • `match_length` - Specifies constraints for matched nets. • `ndr` - Specifies constraints for a net with a non-default rule. • `net_class` - Specifies constraints for shielding a group of nets. It can also be used for a group of nets that have the same NDR. • `shield` - Specifies constraints for shielding a single net. • `simple_net` - Specifies constraints for a simple net. *Data_type*: enum, required

Examples

- The following command sets the layer 1 gap as 0.4, layer 2 gap as 0.39, layer 3 gap as 0.38:
  ```
  set_integration_route_constraint -type diff_pair -net {n_7862 n_7875} -layer_gap {1 0.4 2 0.39 3 0.38}
  ```
- The following command specifies a shield constraint for the `analogS41` net:
  ```
  set_integration_route_constraint -type shield -shield_net VSS -shield_type side -connect_supply anypoint -rule NDR4 -net analogS4I
  ```
- The following command specifies a shield constraint for the net group comprising nets `analogW1I`, `analogW2I`, `analogW3I`, and `analogW4I`:
  ```
  set_integration_route_constraint -type net_class -shield_net VSS -shield_type side -connect_supply anypoint -net {analogW1I analogW2I analogW3I analogW4I}
  ```
- The following command specifies minimum and maximum length constraints for the `xyz` net:
  ```
  set_integration_route_constraint -type simple_net -net xyz -min_length 2.0 -max_length 3.0
  ```
- The following command specifies that individual shields should be used between the nets in the bus constraint:
  ```
  set_integration_route_constraint -type bus -net {nets_1 nets_2 ...} -shield_net VSS -no_share_shield ...
  ```

Related Information

- `delete_integration_route_constraints`
- `get_integration_route_constraint`
- `write_integration_route_constraints`
- Routing Constraint Interoperability chapter in the *Mixed Signal Interoperability Guide*
- Tools Menu chapter in the *Menu Reference*
  - Mixed Signal -> Integration Constraint Editor

write_integration_route_constraints

```
write_integration_route_constraints
[-help]
out_file
[-net net_name | -name string]
```

Writes the constraints in an output file in the following order: `diff_pair`, `match_length`, `bus`, `net_class`, `ndr`, and `shield`.

Notes

- Integration constraints are used to convey the design intent in a mixed-signal design. These constraints are fully interoperable between the digital and custom environments.
- If the design contains one or more complex netclass constraints with net members that have different constraints, `write_integration_route_constraints` reports the following warning:

  ```
  **WARN: (IMPOAX-2124): This design contains unknown complex constraint on complexNetClass
  'contraint_name'. This is not an interoperable routing constraint for Innovus. When
  sourcing the resulting file from this command in Innovus, these complex constraints will be
  lost. You can see the complex constraints in the design in the integration constraint form.
  ```

Parameters

`-help`	Prints a brief description that includes the type and default information for each `write_integration_route_constraints` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man write_integration_route_constraints`
`out_file`	Specifies the output file name that is to be written out. *Data_type*: string, required
`-name` *`string`*	Writes information for specified constraint by name. Use the `get_integration_route_constraint` command to get the name of any constraint in the design. You can specify multiple constraint names, enclosed in braces. *Data_type*: string, optional
`-net` *`net_name`*	Writes the constraint defined on specified net. If the net is part of a netclass, writes constraints for the netclass. You can specify multiple net names, enclosed in braces. *Data_type*: string, optional

Examples

- The following command writes the constraints in the output file called `myfile.rpt`:
  `write_integration_route_constraints myfile.rpt`
  The file `myfile.rpt` could have the following sample constraints that are written out:
  - For `diff_pair` constraint

    ```
    set_integration_route_constraint -type diff_pair -name
    Constr_3__analogNL__DTMF_INST/analogNL_INST -tolerance 20 \
    -rule default \
    -net { \
    DTMF_INST/digsyncc \
    DTMF_INST/digsyncd \
    }
    ```

  - For `match_length` constraint

    ```
    set_integration_route_constraint -type match_length -name
    Constr_4__analogNL__DTMF_INST/analogNL_INST -tolerance 20
    -match_style accordion \
    -rule default \
    -net { \
    DTMF_INST/digsync3 \
    DTMF_INST/digsync4 \
    }
    ```

  - For `net_class` constraint

    ```
    set_integration_route_constraint -type net_class -name
    Constr_11__analogSW__DTMF_INST/analogSW_INST \
    -shield_net VSS -shield_type side -connect_supply anypoint \
    -rule NDR8 \
    -net { \
    analogW2I \
    }
    ```

  - For `shield` constraint

    ```
    set_integration_route_constraint -type shield -shield_net {VSS} -shield_type side \
    -shield_tolerance 30 -net {tdigit[0] tdigit[1] tdigit[2] tdigit[3] tdigit[4] tdigit[5]
    \
    tdigit[6]tdigit[7]}
    ```

- The following command outputs a constraint file called `test.const` for the constraint named `Constr_2__analogNW__DTMF_INST/analogNW_INST`:
  `write_integration_route_constraints test.const -name`

```
Constr_2__analogNW__DTMF_INST/analogNW_INST
```

The output is as follows:

```
set_integration_route_constraint -type diff_pair -name
Constr_2__analogNW__DTMF_INST/analogNW_INST -tolerance 20 \
-rule default \
-net { \
DTMF_INST/cntrlxo[3] \
DTMF_INST/cntrlxo[2] \
}
```

- The following command outputs a constraint file with name `temp.const` for the constraint defined on net `analogW1I`:

  ```
  write_integration_route_constraints temp.const -net analogW1I
  ```

  The output is as follows:

  ```
  set_integration_route_constraint -type net_class -name
  Constr_10__analogSW__DTMF_INST/analogSW_INST \
  -shield_net VSS -shield_type side -connect_supply anypoint \
  -rule NDR8 \
  -net { \
  analogW1I \
  }
  ```

- The following command writes a constraint file named `test1.const` for the specified constraint names:

  ```
  write_integration_route_constraints -name {Constr_2__analogNW__DTMF_INST/analogNW_INST
  Constr_1__analogNW__DTMF_INST/analogNW_INST} -file test1.const
  ```

  The output is as follows:

  ```
  #Integration constraints ascii out for { ADTMF4 dtmf_chip NoConst_10_18 }

  set_integration_route_constraint -type diff_pair -name
  Constr_2__analogNW__DTMF_INST/analogNW_INST -tolerance 20 \
  -rule default \
  -net { \
  DTMF_INST/cntrlxo[3] \
  DTMF_INST/cntrlxo[2] \
  }
  set_integration_route_constraint -type diff_pair -name
  Constr_1__analogNW__DTMF_INST/analogNW_INST -tolerance 5 \
  -rule default \
  -net { \
  DTMF_INST/cntrlxo[0] \
  DTMF_INST/cntrlxo[1] \
  }
  ```

- The following command writes a constraint file named `test2.const` for the specified net names:

  ```
  write_integration_route_constraints -net {DTMF_INST/cntrlxo[3] DTMF_INST/cntrlxo[0]} -file
  test2.const
  ```

The output is as follows:

```
#Integration constraints ascii out for { ADTMF4 dtmf_chip NoConst_10_18 }

set_integration_route_constraint -type diff_pair -name
Constr_2__analogNW__DTMF_INST/analogNW_INST -tolerance 20 \
-rule default \
-net { \
DTMF_INST/cntrlxo[3] \
DTMF_INST/cntrlxo[2] \
}
set_integration_route_constraint -type diff_pair -name
Constr_1__analogNW__DTMF_INST/analogNW_INST -tolerance 5 \
-rule default \
-net { \
DTMF_INST/cntrlxo[0] \
DTMF_INST/cntrlxo[1] \
}
```

Related Information

- delete_integration_route_constraints
- get_integration_route_constraint
- set_integration_route_constraint
- Routing Constraint Interoperability chapter in the *Mixed Signal Interoperability Guide*
- Tools Menu chapter in the *Menu Reference*
    - Mixed Signal -> Integration Constraint Editor

22

Multiple-CPU Processing Commands

- check_multi_cpu_usage
- get_distributed_hosts
- get_multi_cpu_usage
- monitor_distributed_hosts
- report_resource
- set_distributed_hosts
- set_multi_cpu_usage

check_multi_cpu_usage

```
check_multi_cpu_usage
[-help]
```

Checks if distributed processing can work in the software environment. This command also ensures that the working directory is accessible from the master machine as well as the client machines. The `check_multi_cpu_usage` command is specified after configuring the multi-CPU settings (`set_distributed_hosts` and `set_multi_cpu_usage`) to check if all the specified CPUs can be accessed. This command should be invoked before running any commands that use multiple CPUs.

Parameters

`-help`	Outputs the command usage and a brief description about the command.

Examples

Here are some examples.

- The following example command script checks the distributed computing environment when `set_distributed_hosts -local` is specified:

```
> check_multi_cpu_usage


/it/sjclapa06p6_scr02/clre/exp_2017_05_30/0223/dy/2014/2017/0413
/it/sjclapa06p6_scr02/clre/exp_2017_05_30/0223/dy/2014/2017/0413
/it/sjclapa06p6_scr02/clre/exp_2017_05_30/0223/dy/2014/2017/0413
/it/sjclapa06p6_scr02/clre/exp_2017_05_30/0223/dy/2014/2017/0413
Connected to sjfnl624 35779 0 ( PID=13350 )
Connected to sjfnl624 54715 1 ( PID=13356 )
Connected to sjfnl624 40469 3 ( PID=13397 )
Connected to sjfnl624 50442 2 ( PID=13378 )
Task 0 successfully accessed the current working directory.
Task 3 successfully accessed the current working directory.
Task 1 successfully accessed the current working directory.
Task 2 successfully accessed the current working directory.
Task 3: CPU(workload/CPU): 7.84/12 mem(free/total): 94118680k/98813572k.
Task 1: CPU(workload/CPU): 7.84/12 mem(free/total): 94118432k/98813572k.
Task 2: CPU(workload/CPU): 7.84/12 mem(free/total): 94117068k/98813572k.
Task 0: CPU(workload/CPU): 7.84/12 mem(free/total): 94119084k/98813572k.
Test heartbeat for task 0...
The distributed processing environment has been set up correctly.
```

- The following example command script checks the distributed computing environment when when `set_distributed_hosts -lsf -queue lnx64` is specified:

```
> check_multi_cpu_usage

Job <5685326> is submitted to queue <lnx64>. The associated lsf task ID is '0'
Executing
/it/sjclapa06p6_scr02/clre/exp_2017_05_30/0223/dy/2014/2017/0413/.clre_sub_22662_0
Job <5685328> is submitted to queue <lnx64>. The associated lsf task ID is '1'
Executing
/it/sjclapa06p6_scr02/clre/exp_2017_05_30/0223/dy/2014/2017/0413/.clre_sub_22662_1
Job <5685329> is submitted to queue <lnx64>. The associated lsf task ID is '2'
```

```
Executing
/it/sjclapa06p6_scr02/clre/exp_2017_05_30/0223/dy/2014/2017/0413/.clre_sub_22662_2
Job <5685330> is submitted to queue <lnx64>. The associated lsf task ID is '3'
Executing
/it/sjclapa06p6_scr02/clre/exp_2017_05_30/0223/dy/2014/2017/0413/.clre_sub_22662_3
Connected to sjfnl625 43039 1 ( PID=30833 )
Connected to sjfnl625 34830 2 ( PID=30832 )
Connected to sjfnl625 33887 3 ( PID=30831 )
Connected to sjfib046 60125 0 ( PID=32727 )
Task 1 successfully accessed the current working directory.
Task 3 successfully accessed the current working directory.
Task 2 successfully accessed the current working directory.
Task 0 successfully accessed the current working directory.
Task 3: CPU(workload/CPU): 1.43/12 mem(free/total): 74177208k/82277684k.
Task 2: CPU(workload/CPU): 1.43/12 mem(free/total): 74177208k/82277684k.
Task 1: CPU(workload/CPU): 1.43/12 mem(free/total): 74177208k/82277684k.
Task 0: CPU(workload/CPU): 19.79/20 mem(free/total): 352917068k/396858936k.
Test heartbeat for task 0...
The distributed processing environment has been set up correctly.
```

Related Information

- Accelerating the Design Process By Using Multiple-CPU Processing chapter in the *Innovus System User Guide*
  - "Checking the Distributed Computing Environment"
  - set_multi_cpu_usage
  - get_multi_cpu_usage

get_distributed_hosts

```
get_distributed_hosts
{-mode | -hosts | -queue | -resource | -custom_script | -env_script | -args | -
single_cpu_lsf_args | -timeout | -shell_timeout | -report_lsf_info file | -
wait_for_lsf_info value}
[-help]
```

Displays settings for the `set_distributed_hosts` command.

Parameters

`-args`	Displays the additional arguments when the software is running in the Sun Grid Engine (SGE) or Load Sharing Facility (LSF) mode.
`-custom_script`	Displays the name of the custom script when the software is running in `custom` mode.
`-env_script`	Displays the user-defined shell environment script that is run for each of the slave hosts.
`-hosts`	Displays the list of host names when the software is running in `rsh` mode.
`-help`	Outputs the command usage and a brief description about the command parameters.
`-mode`	Displays the distributed processing mode. The possible values are `local`, `sge`, `rsh` , `ssh` , `lsf` and `custom`. *Default*: `local`
`-queue`	Displays the queue name when the software is running in `sge` or `lsf` mode.

`-report_lsf_info` *file*	Specifies to restore the LSF information to the specified file. This parameter can only be used when `set_distribute_host -report_lsf_info` is specified in the current session. You must specify the `get_distributed_hosts -report_lsf_info` only after the LSF tasks exit to obtain the task CPU time. If you specify this command before the task exits, the command gives a warning message.
`-resource`	Displays the resource string when the software is running in `sge` or `lsf` mode.
`-shell_timeout`	Displays the number of seconds the host machine waits for `rsh/ssh` machines to become available for multiple-CPU processing.
`-single_cpu_lsf_args`	Displays the lsf arguments when the software is running in `lsf` mode for a single CPU application.
`-timeout`	Displays the default timeout for a submitted job that will never start.
`-wait_for_lsf_info` *value*	Specifies the timeout period required to wait for task exit and generate the LSF information.

Examples

- The following command displays the current setting for the `-mode` parameter:

  ```
  get_distributed_hosts -mode
  local
  ```

- The following command saves the LSF information to the `lsf.log` file:

  ```
  get_distributed_hosts -report_lsf_info lsf.log
  ```

Related Information

- "Accelerating the Design Process By Using Multiple-CPU Processing" chapter in the *Innovus* System User Guide:
  - Running Distributed Processing
- `set_distributed_hosts`

get_multi_cpu_usage

```
get_multi_cpu_usage
[-help]
[-auto_page_fault_monitor]
[-cpu_per_remote_host]
[-keep_license]
[-local_cpu]
[-remote_host]
[-thread_info]
[-verbose]
```

Displays the requested number of threads or hosts.

To request threads or hosts, use `set_multi_cpu_usage`.

Parameters

`-help`	Outputs the command usage and a brief description about the command parameters.
`-auto_page_fault_monitor`	Displays the warning rank set for performance issues caused by major page faults
`-cpu_per_remote_host`	Displays the number of CPUs available on every remote host for super-threading.
`-keep_license`	Displays the current setting for the `set_multi_cpu_usage -keep_license` parameter.
`-local_cpu`	Displays the number of available local CPUs for multi-threading.
`-remote_host`	Displays the number of available remote hosts for distributed processing or Superthreading.
`-thread_info`	Displays the level of informational messages for each thread.
`-verbose`	Displays a message with the current multiple-CPU setting.

Examples

- The following command displays the requested number of threads for multi-threading:

  ```
  get_multi_cpu_usage -local_cpu
  ```

- The following command displays the requested number of remote hosts for Superthreading or distributed processing:

  ```
  get_multi_cpu_usage -remote_host
  ```

- The following command displays the requested number of CPUs per remote hosts for Superthreading:

  ```
  get_multi_cpu_usage -cpu_per_remote_host
  ```

Related Commands

- set_multi_cpu_usage

monitor_distributed_hosts

```
monitor_distributed_hosts
[-help]
[-log string]
[-overwrite]
[-period integer]
```

Monitors hosts and periodically dumps memory or CPU usage information for each host to a file. This command automatically determines the hosts to be monitored. By default, after every 60 seconds hosts are monitored and the disk information is written to a log file. This command is not required for DMMMC because it also performs client monitoring.

The `monitor_distributed_hosts` command is also supported in DSTA mode.

Parameters

`-help`	Prints out the command usage. For a detailed description of the command and all of its parameters, use the `man` command: `man monitor_distributed_hosts`.
`-log string`	Specifies the log file name. *Default*: host-monitor.log
`-overwrite`	Overwrites the log file, if it already exists.
`-period integer`	Specifies the sampling period after which hosts will be monitored. *Default*: 60

Examples

- A sample monitor log file is shown below:

```
Status keys: E excellent, G good, B bad, T terrible.
Memory values:
total = Real memory of host
progs = Real memory used by processes, including swappable cache
used = Real memory used by processes, excluding swappable cache
Started at: 17/04/25 04:24:22
lgi15(0) TMPDIR is : /tmp/innovus_temp_20836_lgi15_ark_lEAcvw

============ ======== ======================= ========================
Host          CPU      Memory (GB)              TMPDIR (GB)       (Mb/s)
name(id)      util %   total progs used %used   total   avail    rate
```

```
============= ======== ===== ===== ===== ====== ========================
04:24:52
lgi15(0)         6 E     32   24    15   46 E    100.74  94.67 E  262 E
```

Related Commands

- set_multi_cpu_usage

report_resource

```
report_resource
[-help]
[-check_memory_cpu {true | false} [-period integer]]
[-peak]
[-start string | [-end string [-session_cpu] [-diff_memory]]]
[-verbose]
```

Displays the peak memory used during a software session. This command is used to measure the peak memory at any given point in the flow and and make the script better for automation flows.

⚠ The `report_resource` command is only supported in the 64-bit version of the software and Linux port.

Parameters

`-help`	Outputs the command usage and a brief description about the command parameters.
`-check_memory_cpu {true|false}`	Checks the machine's physical memory and CPU resources. When specified, the software enables memory/CPU checking and reports a warning if the free memory is below 5 percent of the total memory or if the average CPU load is larger than 1.35.
`-diff_memory`	Outputs different peak memory and current memory between the step start and step end.
`-peak`	Reports current peak memory till a given point in a flow. This parameter also works with the `-start` and `-end` parameters of the `report_resource` command so that peak memory observed by an individual command can be reported.
`-period integer`	Specifies the period for checking the physical memory and CPU usage. *Default*: 1800s *Min*: 1 *Max*: INT_MAX
`-session_cpu`	Reports the current CPU and real-time after the reports' last step status.

`-start` *`startpoint`*	Specifies the start point to measure the peak memory attained during a specific command. **Note:** The `-start` and `-end` parameters support any user-specified string, such as fullFlow, `time_design`, and TD_PR. You must specify the same string for both the `-start` and `-end` parameters to measure the peak memory attained during a specific command.
`-end` *`endpoint`*	Specifies the end point to measure the peak memory attained during a specific command.
`-verbose`	Gives detailed memory usage information. It displays how much memory is being used at any time and of what form (physical vs. virtual). When specified, the following information is displayed: • `Cpu(s)` is the number of available processors in the machine. • `Load` average is the system load averages for the last 1 minute, 5 minutes, and 15 minutes. • `Mem` and `Swap` are the current memory information of the machine. • `Data Resident Set (DRS)` is the amount of physical memory devoted to other than executable code. "`current mem`" shows this value (Total current DRS) . • `Private Dirty (DRT)` is the memory which must be written to disk before the corresponding physical memory location can be used for some other virtual page. "`peak res`" shows this value (Total peak DRT). This is the minimum number that you must reserve to run the program. • `Virtual Size (VIRT)` is the total amount of virtual memory used by the task. • `Resident Size (RES)` is the non-swapped physical memory a task has used. The number of "Total Peak RES" is the recommended physical memory to reserve.

Examples

- The following command script specifies to display the peak memory attained during the `opt_design` command:

  ```
  report_resource -start "optD"
  opt_design
  report_resource -end "optD"
  ```

  The following message is displayed:

  ```
  Ending "optD" (total cpu=0:15:00, real=0:15:09, peak res=468.8M, current
  mem=396.3M)
  ```

- The following command script specifies to display the peak memory attained during the `time_design` command:

  ```
  report_resource -start fullFlow
  report_resource -start TD_PR
  time_design -post_route
  report_resource -end TD_PR
  report_resource -start OD_PR
  opt_design -post_route
  report_resource -end OD_PR
  report_resource -end fullFlow
  ```

  The following message is displayed:

  ```
  Ending TD_PR (total cpu=0:15:00, real=0:15:09, peak res=468.8M, current
  mem=396.3M)
  Ending OD_PR (total cpu=0:15:00, real=0:15:09, peak res=968.8M, current
  mem=796.3M)
  Ending fullFlow (total cpu=0:35:03, real=0:40:09, peak res=968.8M, current
  mem=796.3M)
  ```

- The following command script specifies to display the peak memory attained during the `opt_design` command:

  ```
  report_resource -peak -start optD1
  report_resource -start optD
  opt_design -post_route
  ```

```
report_resource -end optD
`Ending "optD" (total cpu=0:00:12.9, real=0:00:48.0, peak res=358.8M, current
mem=300.2M)'
report_resource -peak -end optD1
`Ending "optD1" peak res=358.8M'
```

- The following command specifies to display detailed memory information:

  ```
  report_resource -verbose
  ```

  The following output is displayed:

  ```
  Current (total cpu=0:00:23.5, real=0:10:24, peak res=567.8M, current mem=567.8M)

  Cpu(s) 40, load average (past 1 minute): 7.41, load average (past 5 minutes):
  7.46, load average (past 15 minutes): 7.52
  Mem: 527321900k total, 317217944k used, 210103956k free, 3752k buffers
  Swap: 134217724k total, 0k used, 134217724k free, 183952200k cached

  Memory Detailed Usage :
                   Data Resident Set(DRS)  Private Dirty(DRT)  Virtual Size(VIRT)
  Resident Size(RES)
  Total current :   642.3M                 155.6M              1743.3M
   567.8M
  peak :            642.3M                 155.6M              1743.3M
   567.8M
  ```

Related Information

- report_timing

set_distributed_hosts

```
set_distributed_hosts
[-help]
[out_file]
[-add string]
[-args string]
[-custom]
[-custom_script string]
[-custom_script_list string]
[-local]
[-lsf]
[-nc]
[-queue string]
[-remove string]
[-resource string]
[-rsh]
[-shell_timeout integer]
[-single_cpu_lsf_args string]
[-ssh]
[-time_out integer]
[-env_script string]
```

Specifies the multiple-CPU processing configuration for distributed processing or Superthreading. The software finds hosts from the last `set_distributed_hosts` command specified before a distributed processing or Superthreading command starts.

Use `get_distributed_hosts` to display the current settings for the `set_distributed_hosts` command.

ⓘ This command is required for distributed processing and Superthreading.

Parameters

`-help`	Prints out the command usage

out_file	Specifies to report LSF information in a single report. This parameter allows you to collect the system statistics on the status of the distributed jobs. It has the following information: • Host name for master and LSF tasks • Number of CPUs on each machine that are being used by a process • Job ID/PID for each task process • Total memory used on each machine • Total virtual memory used on each machine • Total CPU time used on each machine You must specify this parameter with the `-lsf` parameter. To get the LSF information, specify the `get_distribute_host out_file` command parameter. You must specify the `get_distribute_host out_file` only after the LSF tasks exit to obtain the task CPU time. If you specify this command before the task exits, the command gives a warning message.

`-add` *string*	Adds the specified hosts to the `rsh/ssh` configuration. If you specify more than one host, separate each host name with a space.
	Each time you specify a host, the software uses it to run one client process. To run more than one client process on a host, specify it more than once. For example, if you have two machines that each has two CPUs and you want to use both CPUs in both machines, type a command like the following:
	`set_distributed_hosts -rsh -add {machine1 machine2 \`
	`machine1 machine2}`

`-args` *string*	Specifies any additional arguments you need to provide to the Sun Grid Engine (SGE) or Load Sharing Facility (LSF) job submission command.
`-custom`	Specifies to use the custom job submission script for distributed processing.

`-custom_script` *`string`*	Specifies a custom script to run distributed processing. This includes custom job submission script and arguments for "custom" distributed processing mode. You can use this parameter with `-custom` only.
`-custom_script_list` *`string`*	Specifies custom job submission script list and arguments for "custom" distributed processing mode. for example, `{{{stringCommand1} count1} {{stringCommand2} count2} ... {{stringCommandN} countN}}`. Specifies to customize the launch of multiple jobs, based on resources required for these jobs. This parameter allows you to specify multiple launch strings in order of priority with a maximum allowed job count for each string. If you request for N number of jobs, the command starts submitting jobs starting from *`string1`*, until the corresponding count is reached, at which point it starts submitting the next string in the list. Therefore, the command submits the first "*`count1`*" jobs with *`string1`*, the next "*`count2`*" jobs with *`string2`* and so on. It stops when the total submitted jobs has reached the requested number (N). **Note**: This parameter can only be specified with the `-custom` parameter.
`-env_script` *`string`*	Specifies a user-defined shell environment script that is run for each of the slave hosts. The parameter allows you to pass these environment settings to the slave hosts by including them in the shell environment script. The license variables used by the software are also automatically passed to the slave hosts to avoid license errors.
`-local`	Run distributed processing job on the local machine.
`-lsf`	Specifies a Load Sharing Facility configuration. If you do not specify any parameters for `-lsf`, the software uses default settings. **Note:** To use this parameter, LSF must already be set up (typically by specifying the LSF_ENVDIR and LSF_SERVERDIR environment variables) and bsub must be in your search path. Contact your LSF administrator for details.
`-nc`	Allows you to use NC (Network Computer from RTDA) commands for distributed processing.

`-queue` *string*	Specifies the queue for the LSF or SGE configuration.
`-remove` *string*	Removes the specified hosts or all hosts from the `rsh/ssh` configuration. If you specify more than one host, separate each host name with a space.
`-resource` *string*	Specifies a resource string for the SGE or LSF queue. **Note:** The correct resource string is specific to your installation. Contact your LSF administrator for the appropriate parameters and values.
`-rsh`	Specifies a remote shell configuration. For information, see the `rsh` documentation.
`-shell_timeout` *integer*	Specifies the number of seconds the host machine waits for `rsh/ssh` machines to become available for multiple-CPU processing. *Default:* `5`
`-single_cpu_lsf_args` *string*	Specifies the `bsub` options. For information, see the LSF documentation. **Note:** You must use this parameter to adjust the LSF argument for a single CPU application. Alternatively, use the `-args` parameter to adjust the LSF argument for an application that runs in the Superthreading mode.
`-ssh`	Specifies a secure shell configuration.

`-time_out` *integer*	Specifies the default timeout for a submitted job that will never start. This functionality prevents the master machine from hanging when the clients do not come up for some reason. After timeout, a warning message appears informing you that the distributed CPUs did not come online within the expected time. The master will continue waiting for the clients and it is up to you to continue waiting or kill the run if there are problems with the Distributed Resource Management (DRM) system. A sample warning message is given below: Task id 2 did not come up within the specified timeout period (300 seconds). Use `set_distributed_hosts -timeout` to change the value. This may be a real error due to incorrect or incomplete platform/machine specification provided via set_distributed_hosts to your distributed resource management system (LSF/Sun Grid Engine etc.). If running locally (`set_distributed_hosts -local`), the current machine may be overloaded with too many jobs and swapping (check free memory and CPUs with top). For local runs, consider reducing the number of CPUs you are using via the `set_multi_cpu_usage` command or free up some memory by killing other memory intensive processes. *Default:* 3600 for all the modes (LSF, NC, SGE, RSH, SSH, local and custom **Note**: When the `set_distribute_host` command is not specified, the software sets the default mode as local and the EDP timeout is `3600` seconds.

Examples

rsh

The following command creates an `rsh` configuration and adds hosts `host10`, `host11`, and `host12`:

```
set_distributed_hosts -rsh -add {host10 host11 host12}
```

The following commands create an `rsh` configuration with `host1` and `host2` as remote hosts and then adds `host3` and `host4` to the configuration:

```
set_distributed_hosts -rsh -add {host1 host2}
```

```
set_distributed_hosts -rsh -add {host3 host4}
```

LSF

The following command uses an existing LSF environment, and relies on the general LSF set up by the system administrator. In this example, the name of the queue implies that all the machines have 4 Gb of memory.

```
set_distributed_hosts -lsf -queue mem4G
```

The following command schedules a LSF job named `bigQueue` to run on a Linux machine for a maximum of 30 minutes. No other jobs can run on the machine while `bigQueue` is running.

```
set_distributed_hosts -lsf -queue bigQueue -args {-x -W 30} \
   -resource {OSNAME==Linux}
```

The following command schedules a LSF job named `smallQueue` to run on Linux machines with minimum speed of 2000 MHz.

```
set_distributed_hosts -lsf -queue smallQueue -args {-W 20} \
    -resource {OSNAME==Linux && SPEED>2000}
```

Note: Check with your LSF administrator for the correct values for the resource string, as the parameters and their values differ in each installation. For example, the parameter name for the speed of the machine might be `MHZ` instead of `SPEED` (as in the preceding example), and the units might be specified in gigahertz instead of megahertz.

The following command specifies to report LSF information:

```
set_distributed_hosts -lsf -args "-q lnx64" out_file
get_distribute_host out_file lsf.log
```

Note: Use the above two commands together to save the LSF information in the specified file.

SGE

The following command schedules an SGE job named `all.q` to run on Solaris machines with minimum speed of 2000 MHz.

```
set_distributed_hosts -sge -queue all.q \
-resource {OSNAME==Solaris && SPEED>2000} -args {-A [account name] -N [name]}
```

Local

The following command runs all distributed processing jobs on the machine on which you are running the master version of the software:

```
set_distributed_hosts -local
```

Custom

The following command runs distributed processing with a custom script:

```
set_distributed_hosts -custom -custom_script {bsub -q bigQueue -x -R "OS==Linux && memory>20000"}
```

The following command runs distributed processing with a custom script list:

```
set_distributed_hosts -custom -custom_script_list {{{bsub -P ICD -W 2 -q admin -R "OSNAME==Linux && OSREL==EE50"}1} {{rsh hpc1 -e} 1}}
```

Examples

- The following command allows the client to source the `my_env_file` before start up on the same machine:

  ```
  set_distributed_hosts -local -env_script my_env_file
  ```

  This command allows the master to pass on the user-defined environment settings.

- The following command allows the client to run using the LSF:

  ```
  set_distributed_hosts -lsf -noWaitTaskComeUp
  ```

  When the `-noWaitTaskComeUp` parameter is specified, the software will not wait for all the clients to come up when parts of the clients are launching timeout.

Related Commands

- `get_distributed_hosts`

set_multi_cpu_usage

```
set_multi_cpu_usage
[-help]
[-acquire_license integer]
[-auto_page_fault_monitor {0 | 1 | 2 | 3}]
[-cpu_per_remote_host integer]
[-keep_license {true | false}]
[-license_list string]
[-local_cpu string]
[-release_license]
[-remote_host integer]
[-reset]
[-thread_info {0 | 1 | 2}]
[-verbose]
```

Specifies the number of threads to use for multi-threading, or the maximum number of computers to use for distributed processing, or the maximum number of computers and the number of threads to use for Superthreading. Optionally, reports usage information.

ⓘ This command is required for multi-threading, distributed processing, and Superthreading.

The `set_multi_cpu_usage` command enables multi-threaded timing reporting under the following conditions:

- `timing_report_group_based_mode` attribute is set to `true`. If there are multiple paths with the same slack then the selection can change on multiple runs. This may result in some changes in the `report_timing` output.
- `report_analysis_coverage` command is in non-verbose mode.
- `check_timing -include clock_crossing` parameter is specified.
- `-set_table_style -no_frame_fix_width` parameter is specified.

Parameters

`-help`	Prints out the command usage.

`-acquire_license` *integer*	Acquires licenses to enable the specified number of CPUs. For example, if you specify -acquire_license 5, the software checks out enough licenses to enable 5 CPUs. When specified, the tool displays the number of available CPUs.
	`[DEV]Innovus 1> set_multi_cpu_usage -acquirelicense 5`
	`Total CPU(s) requested: 5`
	`Total CPU(s) enabled with current License(s): 2`
	`Current free CPU(s): 2`
	`Additional license(s) checked out: 1`
	`Encounter_CPU_accelerator_GXL license(s) for 4 CPU(s)`
	`Total CPU(s) now enabled: 6`
	After the licenses are checked out, they can be used for multi-threading, distributed processing, or superthreading.

`-auto_page_fault_monitor {0 | 1 | 2 | 3}`	Specifies to set a warning rank for performance issues caused by major page faults. The parameter supports the following values: • `0` - specifies to disable the reporting of major page faults. • `1` - specifies to set the warning rank as 1. This means `PF/total PF` > 0.8, `b` > 10, and `wa` > 85% • `2` - specifies to set the warning rank as 2. This means `PF/total PF` > 0.5, `b` > 5, and `wa` > 65% • `3` - specifies to set the warning rank as 3. This means `PF/total PF` > 0.2, `b` > 1, and `wa` > 40% where `PF` is the number of major page faults from the software, `total PF` is the total number of major page faults in the machine, `b` is the number of blocked processors, and `wa` is CPU waiting time.
`-cpu_per_remote_host` *integer*	Specifies the number of CPUs on each of the remote machines. This parameter is required for Superthreading. ⓘ For Superthreading, you must use this parameter in conjunction with the -remote_host parameter. *Default:* `1`

`-keep_license {true | false}`	Specifies whether to keep the acquired multiple CPU-licenses until the current session ends. Specify this parameter before running any commands that require multiple-CPU applications. To release all multiple-CPU licenses immediately, use the releaseLicense option. *Default:* `true`
`-license_list` *string*	Specifies the list or order of licenses the software should use for checking out licenses during multiple-cpu processing. The parameter does not support an empty license list {}. You must specify at least one license from the following list: enccpu, nru, edsl, and edsxl.

`-local_cpu` *string*	Specifies the number of CPUs on a local machine. When the `-localcpu` parameter is set to `1` (default), the multi-threaded timing propagation mode is enabled. Also, both CPU multi-threading and distributed process are enabled. The `set_multi_cpu_usage -localCpu` command automatically controls the total number of threads used up to the specified number. The number of specified threads cannot be more than the number of CPUs available in the local computer. Consider a computer having 8 CPUs, then the following command will issue a warning that the number of specified local CPUs is greater than the actual number of CPUs available: `set_multi_cpu_usage -localCpu 10` The local CPU count can be the maximum value that is shown as “processors” on the system under “/proc/cpuinfo”. This value can be obtained by running the “checkSysConf” utility in the "tools/bin" folder. `$> tools/bin/checkSysConf` `============== checkSysConf: Version 3.53 ====================` `OS Version .............: Red Hat Enterprise Linux Workstation release 7.6 (Maipo)` `Kernel Version .........: 3.10.0-957.5.1.el7.x86_64` `Hardware Type ..........: x86_64` `Memory .................: 2063415 Megabyte` `CPU model ..............: Intel(R) Xeon(R) Gold 6254 CPU @ 3.10GHz` **`No. of CPUs ............: 72`** `Total Swap Space........: 131072 Megabyte` `==================================` `$> cat /proc/cpuinfo	grep` **`"processor"`** `| tail -1	awk -F ":" '{print $2 + 1}'` `72` *Default*: 1

`-release_license`	Releases all multiple-CPU license(s) immediately. By default, the software holds multiple-CPU licenses until the end of the current session. To specify that the software release multiple-CPU licenses after every multiple-CPU command runs, use -keep_license.
`-remote_host` *integer*	Specifies the number of remote machines. This parameter is required for distributed processing and Superthreading. *Default:* `0`
`-reset`	Resets the specified parameters back to default values.
`-thread_info {0 | 1 | 2}`	Reports usage information. This parameter works for multi-threading only. *Default:* `0` Specify one of the following values:
	0: Does not write messages to the log file.
	1: Writes the final message to the log file. For example, All threaded jobs finished (10 elapsed sec: 0 processor sec (parent), 0 system sec (parent), 0 processor sec (threads), 0 system sec (threads)).

	2: Writes additional starting/ending information for each thread.
	For example,
	`Starting threaded job 1...`
	`Starting threaded job 2...`
	`Starting threaded job 3...`
	`Ending threaded job 2 (1 elapsed sec, 0 processor sec, 0 system sec, 1.160M).`
	`Ending threaded job 3 (2 elapsed sec, 0 processor sec, 0 system sec, 0.895M).`
	`Ending threaded job 1 (10 elapsed sec, 0 processor sec, 0 system sec, 1.172M).`
	`All threaded jobs finished (10 elapsed sec: 0 processor sec (parent), 0 system sec (parent), 0 processor sec (threads), 0 system sec (threads)).`
	Note: This parameter works in multi-threading mode only.

`-verbose`	Displays messages when changing the multiple-CPU settings.

Examples

To run attribute placement with four threads, specify the following commands:

```
set_multi_cpu_usage -local_cpu 4
place_design
```

To run attribute placement with the maximum number of available threads, specify the following commands:

```
set_multi_cpu_usage -local_cpu max
place_design
```

To run detailed routing with the NanoRoute router in Superthreading mode with two machines and six threads, using an LSF queue, specify the following commands:

```
set_distribute_host -lsf -queue myLSFqueue
set_multi_cpu_usage -remote_host 2 -cpu_per_remote_host 6
route_detail
```

Related Information

- check_multi_cpu_usage
- get_multi_cpu_usage

23

OpenAccess Commands and Attributes

- compare_oa_cellview
- create_oa_lib
- create_oa_lib_property
- get_oa_default_rule_lib
- get_oa_design_data
- oa_read_write Category Attributes
- read_oa
- register_trigger
- report_oa_lib
- set_oa_cell_binding
- unlock_oa_design
- update_oa_lib
- write_lef_library
- write_oa
- write_oa_abstract
- write_oa_black_boxes
- write_oa_techfile

compare_oa_cellview

```
compare_oa_cellview
[-help]
[-ignore {pin_purpose}]
[-out_file report_file_name]
{ {{-source {from_lib from_cell from_view} -compare {to_lib to_cell to_view}} | {-lib lib -cell cell -view
{sourceView compareView}}} }
```

Compares specified cell views. Use this command to compare two OpenAccess databases. For instance, you can use this command to compare the lib/cell/view specified using `-source` to the lib/cell/view specified with `-compare`. Boundary and terminals (name, direction, pin location and size) are compared. This command can be used without design import.

Parameters

`-help`	Prints a brief description that includes type and default information for each `compare_oa_cellview` parameter. For a detailed description of the command and all of its parameters, use the man command: `man compare_oa_cellview`.
`-cell cell`	Specifies the cell for comparing two views. *Data_type*: string, optional
`-compare {`*`to_lib to_cell to_view`*`}`	Specifies the library/cell/view that is to be compared. *Data_type*: string, optional
`-ignore {pin_purpose}`	Specifies the list of attributes to be excluded from the cell comparison. As of now, the only attribute that can be ignored is the purpose, which is specified using the value `pin_purpose` with `-ignore`. *Data_type*: (enum)+, optional
`-lib` *`lib`*	Specifies the library for comparing two views. *Data_type*: string, optional
`-out_file` *`report_file_name`*	Specifies the report file in which all the comparison detail is to be written. *Data_type*: string, optional
`-source {`*`from_lib from_cell from_view`*`}`	Specifies the source library, cell, and view. *Data_type*: string, optional
`-view {`*`sourceView compareView`*`}`	Specifies different views of the same library/cell. *Data_type*: string, optional

Examples

- The following command compares two views of the same cell that are present in the same library `zambezi45` and writes the report to the `test1a` file:

  ```
  compare_cellview -source {zambezi45 pll_vcodiv layout} -compare {zambezi45 pll_vcodiv abstract} -out_file
  test1a
  ```

  A sample output is displayed below:

  ```
  ################################################
  # #
  # Compare_cellview comparison #
  # #
  ################################################

  Source Design : 'zambezi45/pll_vcodiv/layout' is identified as layout design
  Compare Design : 'zambezi45/pll_vcodiv/abstract' is identified as abstract design

  Source Design seems to have been modified after the Compare Design has been created

  Compare Celltype

  Source Design has celltype 'none' & Compare Design has Celltype 'block'

  Compare Cell Boundary

  Source Design has different PRBoundary than Compare Design

  Compare Terminals

  Source Design has Pin 'en' which has Port with shape (36.215 0 36.355 0.14) on layer 'Metal5' and purpose
  'pin' is not present in Compare Design
  Compare Design has Pin 'en' which has a Port with shape (36.215 0 36.36 0.14) on layer 'Metal5' and purpose
  'net' is not present in Source Design
  ...

  Antenna Information

  Antenna Information is defined for Pin 'en' in Compare but not defined in Source Design
  Antenna Information is defined for Pin 'vcocalen' in Compare but not defined in Source Design
  ...

  Source Design is not equivalent to Compare Design
  ```

- The following command compares the views `fplan1` and `fplan2` of the library `DesignLib` and the cell `myAlu`:

  ```
  compare_oa_cellview -source {DesignLib myAlu fplan1} -compare {DesignLib myAlu fplan2}
  ```

create_oa_lib

```
create_oa_lib
[-help]
oa_lib
[-out_dir path]
[-no_compress]
{-oa_attach_tech_lib techlib |
-oa_copy_tech_lib techlib |
-oa_reference_tech_libs techliblist}
```

Copies or attaches a technology database to a library. If the library exists, this command does not update the library.

Note: The parameters must appear in the order specified by the syntax.

The `create_oa_lib` command is part of the OpenAccess design flow. Use this command to create an OpenAccess library at the specified path. `create_oa_lib` needs to be called only once before the design is saved. Subsequent `write_db` calls to the same library will automatically find out the library path that gets stored in `cds.lib`.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `create_oa_lib` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man create_oa_lib`.
`oa_lib`	Specifies the name of a library to be created. The software creates the library and adds its name to the `cds.lib` file. *Data_type*: string, required
`-no_compress`	Creates a new library without any compression irrespective of the compression settings provided by the `oa_new_lib_compress_level` attribute. *Data_type*: bool, optional
`-oa_attach_tech_lib` *`techlib`*	Attaches the technology database specified by the technology library to the library that is to be created. *Data_type*: string, required
`-oa_copy_tech_lib` *`techlib`*	Copies the technology database specified by *`techLib`* to the library specified by *`oa_lib`*, and saves the library to the current working directory. *Data_type*: string, required
`-oa_reference_tech_libs` *`techliblist`*	Creates a local technology database in the design library that can be modified incrementally. *Data_type*: string, required
`-out_dir` *`path`*	Specifies the name of a directory in which to save the library. The path you specify must include the directory that will become the library itself. *Default:* *`./oa_lib`* (Saves the library to the current working directory) *Data_type*: string, optional

Examples

- The following command creates the library `myLib` in the current directory, and attaches the technology database in `myRefLib` to `myLib`:
  ```
  create_oa_lib myLib -oa_attach_tech_lib myRefLib
  ```
  Note: Note: If `myRefLib` is read-only, the `write_db -oa_lib_cell_view` command generates an error if it needs to update the technology database.
- The following command creates the library `TEST_LIB` at the specified path, and attaches the technology database in `TEST_REF_LIB` to TEST_LIB:
  ```
  create_oa_lib TEST_LIB -out_dir ./outputs/TEST_LIB -oa_attach_tech_lib TEST_REF_LIB
  ```
- The following command generates an error because only one reference library can be specified:
  ```
  create_oa_lib myLib -oa_attach_tech_lib {myRefLib1 myRefLib2}
  ```
- The following command creates the library `myLib`, and copies the technology database from `myRefLib` to `myLib`. If `myRefLib` is modified, the two technology databases will become out of sync. Additionally, any future write operations from the `write_db -oa_lib_cell_view` command will modify the technology database copy in `myLib` and also cause the databases to be out of sync.
  ```
  create_oa_lib myLib -oa_copy_tech_lib myRefLib
  ```
- The following command creates a new library which references existing technology databases (`refLib1` and `refLib2`), but allows local technology updates if required (typically to add new custom vias to the local technology database).
  ```
  create_oa_lib myLib -oa_reference_tech_libs {refLib1 refLib2}
  ```

Related Information

- `create_oa_lib_property`
- `oa_read_write` Category Attributes
  - `oa_new_lib_compress_level`
- `write_db`

create_oa_lib_property

```
  create_oa_lib_property
[-help]
-name prop_name
-oa_lib libName
-type {boolean integer float string}
-value propValue
```

Adds a new property to the specified OpenAccess library. This command cannot be used to change the value of an existing property.

Note: You cannot use the name `techLibName` for the property you are adding because this name is used by the OpenAccess infrastructure to store the property attached to the library.

Parameters

`-help`	Prints a brief description that includes type and default information for each `create_oa_lib_property` parameter. For a detailed description of the command and all of its parameters, use the man command: `man create_oa_lib_property`.
`-name` *`prop_name`*	Specifies the name of the property being created. *Data_type*: string, required
`-oa_lib` *`libName`*	Specifies the name of the OpenAccess library to which the property has to be added. *Data_type*: string, required
`-type {boolean integer float string}`	
	Specifies the type of the property being created. *Data_type*: enum, required
`-value` *`propValue`*	Specifies the value to be set for property being created. The value is interpreted from the input string and converted to the specified type. *Data_type*: string, required

Example

In Innovus, the following set of commands creates a library named *`myLib`*, adds the *`prop1`* property to it, and saves the design before exiting:

```
create_oa_lib myLib -oa_attach_tech_lib adiTech

create_oa_lib_property -oa_lib myLib -name prop1 -type string -value value1

write_db -oa_lib_cell_view {myLib block1 layout}

exit
```

You can now add more properties to the library in Virtuoso. When you restore the design in Innovus, all properties previously added to the library, whether in Innovus or Virtuoso, reflect correctly.

Related Information

- create_oa_lib

get_oa_default_rule_lib

```
get_oa_default_rule_lib
[-help]
[-rule rule_name]
[-libs lib_list]
[-verbose]
```

Searches through a specified library (or libraries) to check whether a specified LDRS exists or not. You can run this check before reading a library.

Parameters

`-help`	Prints a brief description that includes type and default information for each `get_oa_default_rule_lib` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man get_oa_default_rule_lib`
`-libs lib_list`	Specifies the list of library names to search through. If not specified, all libraries are searched. *Data_type*: string, optional

`-rule rule_name`	Specifies the name of the rule to find.
	If not specified, the `init_oa_default_rule` attribute is used. If the attribute has an empty string value, the standard lookup (`LEFDefaultRouteSpec` by type, then by name) is used.
	If the specified *rule_name* does not exist, the command displays an error such as:
	`Rule '<rule_name>' was not found in specified libraries.`
	If the specified *rule_name* is found, the command returns the name of the library where the specified *rule_name* was found. This could be from the `-libs lib_list` or any ITDB that one of those libraries refers to.
	If the specified *rule_name* is found but is not a legal LDRS type as determined by the OAX reader code, the command displays an error such as:
	`Constraint group '<rule_name>' was found in library '<lib_name>', but is not a legal LEFDefaultRouteSpec`
	Data_type: string, optional

`-verbose`	Echoes details of library name where the rule was found (including descending through technology references from libraries specified in the *lib_list*. *Data_type*: bool, optional

Examples

- The following example shows how `get_oa_default_rule_lib` can be used to search for a specific library and the graph of techs that it references:

```
set_db init_oa_ref_libs "tech_lib macro_lib"
set_db init_oa_default_rule LDRS_65nm_6lm
set tech_lib [lindex $init_oa_ref_libs 0]
set lib_found [get_oa_default_rule_lib -libs $tech_lib] # use the init_oa_default_rule to look up the library
if {$lib_found != ""} {
Puts "$init_oa_default_rule was found in $lib_found or in the graph of libraries that $tech_lib references"
} else {
Puts "$init_oa_default_rule was not found in $tech_lib nor in the graph of libraries that $tech_lib
references"
}
```

- The following command shows how the `-verbose` option can be used:

```
get_oa_default_rule_lib -verbose
INFO: Default rule 'LEFDefaultRouteSpec' was found in library 'FEOAreflib8'.
FEOAreflib8
```

Related Information

- `init` Category Attributes
  - `init_oa_default_rule`
  - `init_oa_ref_libs`

get_oa_design_data

```
get_oa_design_data
[-help]
{{-all_oa_libs} | {{-design_data dd_objHandle | -oa_lib libName | -oa_cell {lib cell} | -oa_cellview {lib cell
view}}
[-children | -file | -path | -type |-name]}}
```

Returns OpenAccess library information.

Parameters

`-all_oa_libs`	Returns a list of design data (`dd`) object handles for the list of libraries present in the library definition file (`cds.lib`). *Data_type*: bool, required
`-children`	Returns the `dd` object handles of the children of the specified `dd` object handle. Children of libraries are cells. Children of cells are views. *Data_type*: bool, optional
`-design_data dd_objHandle`	Specifies the `dd` object to be processed. *Data_type*: string, required
`-file`	Returns the `dd` object handles of all the files in the specified `dd` object handle. *Data_type*: bool, optional
`-name`	Returns the name of the specified `dd` object handle. *Data_type*: bool, optional
`-oa_cell {lib cell}`	Specifies the names of the library and cell to be processed. *Data_type*: string, required
`-oa_cellview {lib cell view}`	Specifies the names of the library, cell, and view to be processed. *Data_type*: string, required
`-oa_lib libName`	Specifies the name of the library to be processed. *Data_type*: string, required
`-path`	Returns the absolute path name (`dd` path) of the specified `dd` object handle. *Data_type*: bool, optional
`-type`	Returns the type of the specified `dd` object handle. *Data_type*: bool, optional

Examples

- The following command returns the path of the specified library/cell/view as shown below:

  ```
  get_oa_design_data -cellview {pdkLib cell1 abstract} -path
  ```

  returns:

  ```
  /libs/pkdLib/cell1/abstract
  ```

  This command is equivalent to the following SKILL function:

  ```
  ddGetObjReadPath(ddGetObj("pdkLib" "cell1" "abstract"))
  ```

- The following proc traverses the cds.lib structure (expanding #INCLUDE as required) to get the complete list of OpenAccess libraries and their full paths.

  ```
  # expand #INCLUDE as needed and print all libraries and their full paths
  proc print_all_lib_paths {} {
     puts [format "%16s -> %s" "Library Name" "Library Path"]
     puts "------------------------------------------------------------------"
     foreach oa_lib [get_oa_design_data -all_oa_libs] {
         puts [format "%16s -> %s" [get_oa_design_data -design_data $oa_lib -name] [get_oa_design_data -
  design_data $oa_lib -path]]
     }
  }
  ```

oa_read_write Category Attributes

The root attributes in the `oa_read_write` category are:

- oa_allow_analysis_only
- oa_allow_bit_connection
- oa_allow_tech_update
- oa_bindkey_file
- oa_cell_view_dir
- oa_convert_diagonal_path_to
- oa_cut_rows
- oa_display_resource_file
- oa_display_resource_file_in_library
- oa_drc_fill_purpose
- oa_enclose_quick_abstract_pins
- oa_full_layer_list
- oa_full_path
- oa_inst_placed_if_none
- oa_lib_create_mode
- oa_lock
- oa_logic_only_import
- oa_merge_trim
- oa_new_lib_compress_level
- oa_pin_purpose
- oa_push_pin_constraint
- oa_quick_abstract_for_custom_cells
- oa_read_system_reserved_layers
- oa_silently_ignore_unsupported_vias
- oa_text_purpose
- oa_tie_net
- oa_update_mode
- oa_use_virtuoso_bindkey
- oa_use_virtuoso_color
- oa_view_sub_type
- oa_write_cds_fixed_vias
- oa_write_mask_data_locked
- oa_write_net_voltage

- oa_write_relative_path

To set the value of any root attribute, use the `set_db` command:

```
set_db attribute_name value
```

To get the current value of any root attribute, use the `get_db` command:

```
get_db attribute_name
```

To get the current value of all attributes in the `oa_read_write` category, use the following command:

```
get_db -category oa_read_write
```

oa_allow_analysis_only

`oa_allow_analysis_only {true | false}`

Default: `false`

Data_type: bool, read/write

Allows analysis only flow.

`assemble_design` is not allowed when `oa_allow_analysis_only` is set to `false` and `oa_update_mode` is set to `true`, thereby locking the OpenAccess databse. To allow `assemble_design` to proceed, you must change `oa_allow_analysis_only` to `true`. If you set `oa_allow_analysis_only` to `true` and the Innovus session is in `oa_update_mode` = `true` mode (whether set explicitly or internally on the fly), the `oa_update_mode` setting is ignored.

Note that there may be loss of some semantic information, such as MPPs, while saving the design when `oa_allow_analysis_only` is set to `true`. However, the saved design would be geometrically correct.

oa_allow_bit_connection

`oa_allow_bit_connection {true | false}`

Default: `false`

Data_type: bool, read/write

Creates databases for instances that only have bit-based terminals, without requiring the corresponding bus terminals.

When set to `true`, it saves the connectivity information by bits if the bus connections are not available.

When set to `false`, an error will be issued if the abstract view is missing expected bus terminal information.

oa_allow_tech_update

`oa_allow_tech_update {true | false}`

Default: `true`

Data_type: bool, read/write

Allows the technology database to be updated. The common reason that a technology database would need to be updated is that new vias have been added to the design that are not in a parameterized form. Via masters that are a simple list of rectangles require that a "custom via" be created in the technology database, If the design library had been created with an attached technology, the value should be set to `false` allowing a reference library technology that is shared by multiple designs to have design specific vias to be added to it. For design libraries that have been created in reference mode, allowing the technology database to be updated is typically fine but normally it is best to always use paramaterized vias for OpenAccess-based flows.

Using `set_db generate_special_via_symmetrical_via_only true` before any power planning or routing will typically prevent the creation of non-parameterized vias in the flow.

When `oa_allow_tech_update` is set to `false`, the creation of the OpenAccess database will fail if any custom vias would need to be created within the design library.

Related Commands and Attributes

`generate_special_via_symmetrical_via_only`

oa_bindkey_file

`oa_bindkey_file` *`fileName`*

Default: `""`

Data_type: string, read/write

Maps into Innovus equivalent bindkeys. If the bindkey file is not specified, the Virtuoso default bindkey information is used.

For more information, see the "SKILL to TCL Mapping" section in the Importing and Exporting Designs Design Import and Export in Stylus chapter of the *User Guide*.

Related Commands and Attributes

`oa_use_virtuoso_bindkey`

oa_cell_view_dir

`oa_cell_view_dir` *`string`*

Default: ""

Data_type: string, read/write

Specifies the path name to the physical location of the lib/cell/view that was restored using `read_db -oa_lib_cell_view`. `oa_cell_view_dir` is a system-defined, transient (non-persistent) attribute used in the OpenAccess flow. This should be treated as a *read-only* attribute.

Related Commands

`read_db`

oa_convert_diagonal_path_to

`oa_convert_diagonal_path_to {polygon | default_style}`

Default: `polygon`

Data_type: enum, read/write

Enables interoperability of the end style of geometries of type path segment between Virtuoso and Innovus.

This attribute applies to the diagonal path segments only.

- If the end-style of a diagonal path segment is `custom`, then the end style is retained as is.

- If the end style of a diagonal path segment is not `custom`, then if you set the value to `polygon`, Innovus converts the path segment into a geometrically equivalent polygon.
- If the value specified is `default_style`, then the end style of the path segment is set to `default_style`. In this case, Innovus retains the semantics of the object to be a path segment.

Important: To ensure that there is no loss of semantic or geometric information, the path segments in Virtuoso -XL and Virtuoso-GXL should use the `custom` end-style on diagonal path segments which can be created using *Create-Shape-Geometric Wire*.

oa_cut_rows

```
oa_cut_rows {true | false}
```

Default: `false`

Data_type: bool, read/write

Cuts rows against placement obstructions, block halos, or power domain fence minimum gaps.

The following command cuts rows against placement obstructions, block halos, or power domain fence minimum gaps:

```
set_db oa_cut_rows true
```

oa_display_resource_file

```
oa_display_resource_file pathName
```

Default: `""`

Data_type: string, read/write

Allows you to specify a specific `display.drf` file that is to be used. It overrides all the searching options and the specified file is used.

The following command searches for the `mydisplay.drf` file irrespective of the value of the `oa_display_resource_file_in_library` attribute:

```
set_db oa_display_resource_file /servers/oastore/mydisplay.drf
```

oa_display_resource_file_in_library

```
oa_display_resource_file_in_library {true | false}
```

Default: `false`

Data_type: bool, read/write

Allows the tool to search for the `display.drf` file in the technology library (or complete technology graph in case of ITDB) when set to `true`. If not found, it looks for the file in the current working directory followed by the home directory.

When set to `false`, the tool searches for the `display.drf` file first in the current working directory followed by the home directory.

oa_drc_fill_purpose

```
oa_drc_fill_purpose {gap_fill | drawing}
```

Default: `gap_fill`

Data_type: enum, read/write

Specifies whether to save DRCFILL shapes to the `gap_fill` or `drawing` purpose. By default, DEF routes with + SHAPE DRCFILL are mapped to the `gapFill` purpose in OpenAccess. However, some OpenAccess tech files do not have this purpose defined so any shapes mapped to the `gapFill` purpose would not be visible in Virtuoso and would not be written out in GDS or OASIS. In such cases, set the `oa_drc_fill_purpose` attribute to `drawing` to save these shapes to the `drawing` purpose. When you do so, these shapes get an additional internal attribute so they can round-trip back to DRCFILL shapes properly.

oa_enclose_quick_abstract_pins

`oa_enclose_quick_abstract_pins {true | false}`

Default: `true`

Data_type: bool, read/write

Specifies that shapes enclosing the layout pin shapes should be included as part of the pins during Quick Abstract Inference (QAI) pin modeling. Check the Quick Abstract Inference chapter in the *Mixed Signal (MS) Interoperability Guide* for details.

oa_full_layer_list

`oa_full_layer_list {true | false}`

Default: `true`

Data_type: bool, read/write

Imports the complete list of OpenAccess layers, when importing an OpenAccess design.

If you specify `false`, only OpenAccess layers defined with `oacMaterial` and type are imported.

oa_full_path

`oa_full_path {true | false}`

Default: `false`

Data_type: bool, read/write

Creates the `DEFINE` statements in the `lib.def` files using full paths. If you specify `-oa_full_path false`, the software creates the statements using relative paths.

The following command instructs the Innovus software to create the `DEFINE` statements using full paths:

```
set_db oa_full_path true
```

oa_inst_placed_if_none

`oa_inst_placed_if_none {true | false}`

Default: `true`

Data_type: bool, read/write

Maps the placement status of instances in the OpenAccess database to the placement status in the Innovus software. When set to `true`, the instances in the OpenAccess database that have an unknown placement status are treated as "placed" in the Innovus software. When set to `false`, instances with an unknown placement status are treated as "unplaced."

oa_lib_create_mode

```
oa_lib_create_mode {none | attach | copy | reference}
```

Default: `reference`

Data_type: enum, read/write

Specifies the mode in which `write_db -oa_lib_cell_view` creates a library.

- `none`: Turns off the ability to have `write_db` create a library if it does not already exist.
- `attach`: Attaches the technology database specified by the technology library to the library that is to be created.
- `copy`: Copies the technology database specified by `techLib` to the library specified by `libName`, and saves the library to the current working directory.
- `reference`: Creates a local technology database in the design library that can be modified incrementally. This is the recommended setting to handle the case where the design may have "custom vias" added at some point by power planning/routing.

Related Commands

- `write_db`

oa_lock

```
oa_lock {true | false | error_if_locked}
```

Default: `false`

Data_type: enum, read/write

Specifies whether or not the OpenAccess database will be locked when the design is restored in Innovus. You can specify one of the following options:

- `false`: This is the default option. When set to `false`, the tool does not check whether the design cell view is locked before restoring the design. The OpenAccess database can continue to be edited by other processes even while the design is restored in Innovus.
  Note: If the `oa_update_mode` attribute is set to `true`, the software automatically locks the database, even if the `oa_lock` attribute is set to `false`.
- `true`: When set to `true`, Innovus locks the design cell view during `read_db`, provided that it is not already locked by some other tool. With this setting, other tools cannot edit the OpenAccess database as long as it is locked for the Innovus session but they can still access the database in the read-only mode. To remove the lock, specify the `reset_design` command or exit the Innovus software.
  Note: If the design cell view is already locked by some other tool before `read_db`, Innovus issues a warning but continues to read in the cell view.
- `error_if_locked`: If you want the tool to be more restrictive, set the `oa_lock` attribute to `error_if_locked`. With this option, if the design is already locked while importing or restoring a design, the tool aborts and reports the following error:

  ```
  **ERROR: (IMPOAX-7995): updateMode/locking is on. Design is already locked by another tool. You are
  recommended to close this design from another application before proceeding.
  ```

Default: `false`

ⓘ The value of `-locking` cannot be changed while a design is in memory.

Notes

- If you set the `oa_update_mode` attribute to `true`, the software automatically locks the database, even if the `oa_lock` attribute is set to `false`.
- The value of `oa_lock` cannot be changed while a design is in memory.

Related Commands

- `unlock_oa_design`

oa_logic_only_import

```
oa_logic_only_import {true  | false}
```

Default: `false`

Data_type: bool, read/write

Controls what information the software reads when importing an OpenAccess maskLayout database.

When set to `true`, the software reads only the logical (Verilog netlist equivalent) connectivity during the `init_design` process. When set to `false` (the default value), the software automatically reads physical information, such as floorplan, placement, and routing, from the database, and updates physical net connectivity for power and ground connections.

Note: The `oa_logic_only_import` attribute applies only to OpenAccess maskLayout databases. For other types of OpenAccess databases, the software always reads only the logical connectivity.

Related Commands

- `init_design`

oa_merge_trim

```
oa_merge_trim {true  | false}
```

Default: `true`

Data_type: bool, read/write

Merges the trim metal shapes during save when the trim metal gap is less than or equal to the max_gap settings defined with the `design_merge_trim_shapes` attribute. Refer to the `design` category attribute, `design_merge_trim_shapes`, for detailed merging options.

Related Commands

- `design`
- `write_oa`

oa_new_lib_compress_level

`oa_new_lib_compress_level` *`value`*

Default: `1`

Data_type: int, read/write

Sets the compression level of a library during the library creation process. Cellviews written to a library will follow the compression level of the library and not the current `oa_new_lib_compress_level` setting.

Valid values range between 0 to 9. A value of 0 indicates that compression is turned off.

Related Commands

- `create_oa_lib`

oa_pin_purpose

`oa_pin_purpose {true | false}`

Default: `false`

Data_type: bool, read/write

Writes out pin shapes with the purpose `pin`. When set to `false`, writes out pin shapes with the purpose `drawing`.

oa_push_pin_constraint

`oa_push_pin_constraint {true | false}`

Default: `false`

Data_type: bool, read/write

Specifies whether or not `write_db` or `write_oa_black_boxes` should push an interoperable pin constraint into its corresponding blackbox abstract cellview.

Note: Interoperable pin constraints refer to pin constraints that are understood by Innovus, Virtuoso, and PVS.

Related Commands

`write_db`

`write_oa_black_boxes`

oa_quick_abstract_for_custom_cells

`oa_quick_abstract_for_custom_cells {pcells stdcells all off}`

Default: `all`

Data_type: enum, read/write

Trims design shapes of custom cells. By default, this capability is turned on for all types of custom cells. The following settings are allowed for this attribute:

- `pcells` - Trims design shapes of only custom parameterized cells (PCells).

- `stdcells` - Trims design shapes of only custom standard cells.
- `all` - Trims design shapes of all custom cells.
- `off` - Does not trim design shapes of any custom cell.

Note: This feature needs the INVS30 (Innovus Mixed Signal Option) license. If this license is not available, the abstract that gets generated for custom cells will not work properly in Innovus.

oa_read_system_reserved_layers

```
oa_read_system_reserved_layers  {all | text | off}
```

Default: `all`

Data_type: enum, read/write

Reads the system-reserved layers from the PDK. The following settings are supported:

- `all` - All system-reserved layers are read.
- `text` - Only text layer is read.
- `off` - No system-reserved layer is read.

oa_silently_ignore_unsupported_vias

```
oa_silently_ignore_unsupported_vias  {true | false}
```

Default: `false`

Data_type: bool, read/write

Specifies whether `read_db` should issue messages about vias that are not supported in Innovus when a Virtuoso design is read in.

oa_text_purpose

```
oa_text_purpose purpose_name
```

Default: `drawing`

Data_type: string, read/write

Specifies the purpose name that will be used when text labels are written.

By default, any text label created in Innovus (typically via the `create_text` command) is automatically mapped to the `drawing` purpose. Use the `oa_text_purpose` attribute to map the text labels to a user-defined purpose.

Note: `oa_text_purpose purpose_name` overrides the default `drawing` purpose on output.

oa_tie_net

```
oa_tie_net {tieHighNetName tieLowNetName}
```

Default: `""`

Data_type: string, read/write

Specifies the global tie high and tie low net names to be created in the OpenAccess database. The nets are used to represent the Verilog `1'b1/1'b0` connections to the instance ports. Typically, `tie1` and `tie0` are used as tie high and tie low net names,

respectively. If these names exist in the design locally in any module, then the global names should be changed.

oa_update_mode

`oa_update_mode {true | false | auto}`

Default: `false`

Data_type: enum, read/write

Specifies whether or not to save or restore an OpenAccess design in update mode. If `oa_update_mode` is set to `true`, `read_db -oa_lib_cell_view` processes parameterized cells (PCells) and other custom objects correctly in the Innovus database. For information about generating/regenerating the PCell cache, refer to the "Preparing the IP Library" section in the Technology Data Preparation chapter of the *Mixed Signal (MS) Interoperability Guide*. Setting

`oa_update_mode` to `true` also ensures that `write_db -oa_lib_cell_view` updates the Innovus-modified portion of an OpenAccess database without making changes to PCells or other custom objects.

⚠ In the `oa_update_mode true` mode, the tool locks the source cellview for the duration of the Innovus session.

The `oa_update_mode` attribute provides the ability to import OpenAccess database created from Virtuoso that contains objects which are not understood by Innovus. Complete the digital part in Innovus and update only the objects modified by Innovus in the original OpenAccess database. If your design has any of the following objects, they would only be visible (not editable) in Innovus:

- PCells
- Modgen cells
- Fig-groups
- PCell vias
- RODs (Example, Multi-Part Paths, and so on)
- User defined properties on design objects such as nets, shapes (`lxStickyNet`), instances, abstract views, intermediate hierarchical instances, and so on.

You won't be allowed to edit or modify these objects.

On using the `oa_update_mode` attribute, the property on shapes (for example, `lxStickyNet`) and nets created in VLS-XL / GXL would be kept as-is.

In case of designs where parameterized vias are used, Innovus understands these attributes and evaluates them on-the-fly to create an exact shape and orientation of the cuts.

If the `oa_update_mode` attribute is set to `auto` and the cellview contains objects that are not supported in Innovus, the software internally enables update mode with a message such as follows:

```
INFO: Objects have been found in the cellview requiring updateMode to be enabled so that there is no loss of semantic information when the cellview is saved.
```

Note that enabling update mode can affect `write_db` performance.

Notes

- The value of the `oa_update_mode` attribute cannot be changed while a design is in memory. If this attribute has been set to `true` before importing an OpenAccess design into Innovus, you cannot convert the `oa_update_mode` attribute to `false` in any new Innovus session.
- The on-disk OpenAccess cellview is locked between the restore and save steps when the `oa_update_mode` attribute is set to `true`.

- In `oa_update_mode auto`, the lock is applied at the time that `read_oa` or `read_db -oa_lib_cell_view` starts and released at the end of the command if no objects were encountered that required updateMode to be enabled.

oa_use_virtuoso_bindkey

`oa_use_virtuoso_bindkey {true | false}`

Default: `false`

Data_type: bool, read/write

Allows you to use Virtuoso bindkeys.

oa_use_virtuoso_color

`oa_use_virtuoso_color {true | false}`

Default: `false`

Data_type: bool, read/write

Allows you to display colors and stipples for the design layers (including full custom layers). Using this attribute, Innovus utilizes the same colors and stipples that you would see if you open the design in Virtuoso.

oa_view_sub_type

`oa_view_sub_type {vce | vxl | none}`

Default: `vxl`

Data_type: enum, read/write

Indicates which Virtuoso application should be used for viewing the layout cellviews in the saved OpenAccess database.

- When set to `vce`, cellviews are opened in Virtuoso Chip Editor.
- When set to `vxl`, cellviews are opened in Virtuoso XL.
- When set to `none`, it indicates that the OpenAccess database can be edited by any editor within Virtuoso platform based on its license availability.

oa_write_cds_fixed_vias

`oa_write_cds_fixed_vias {new_vias_only | off}`

Default: `off`

Data_type: enum, read/write

Controls how fixed/custom vias are created in OpenAccess library when saving a design from Innovus to OpenAccess:

- `off` - No cdsFixedVias are created. All fixed/custom vias in the design are saved as oaCustomVia.
- `new_vias_only` - Any newly created custom via is saved as a fixed via while any pre-existing custom via in the Incremental Technology Database (ITDB) is reused when saving the design.
  Note: This feature needs the INVS30 (Innovus Mixed Signal Option) license. If this license is not available, the default `off` value will be used and no fixed vias will be created.

oa_write_mask_data_locked

```
oa_write_mask_data_locked {true | false}
```

Default: `false`

Data_type: bool, read/write

Sets the shapes with mask coloring (mask values other than 0) to locked in the saved cellview, when set to `true`.

By default, `write_oa` and `write_db` write out shapes with their mask coloring information in an unlocked state. Some tools treat the unlocked color information as the recommended color. However, for some foundry processes, shapes with unlocked mask coloring information are written to GDSII/Stream as gray (uncolored). In such cases, set the `oa_write_mask_data_locked` parameter to `true` to lock the mask coloring of shapes explicitly.

Note: This option affects only the saving of shapes with non-zero mask coloring values. Review the map file that will be used to create GDSII/Stream output from the OpenAccess cellview to determine if the map file requires shapes to be locked for them to be written out as colored.

Related Commands

- `write_db`
- `write_oa`

oa_write_net_voltage

```
oa_write_net_voltage {true | false}
```

Default: `false`

Data_type: bool, read/write

Saves the minimum and maximum net voltage across all loaded timing views to every net. When this attribute is set to `true`, `write_db` writes voltage information to the voltage attribute of oaBitNet. This is useful for Virtuoso, and other tools that may read the cellview, to understand the minimum and maximum net voltage without needing to interpret the CPF or .lib file to get that information. The round-trip of the information into Innovus is not dependent on this setting.

oa_write_relative_path

```
oa_write_relative_path {true | false}
```

Default: `true`

Data_type: bool, read/write

Specifies whether or not to save the relative path in the configuration file. Defines the current working directory (`$cwd`) in the configuration file as ". ". When set to `true`, a design can be restored, even if the directory path of the design file is changed.

read_oa

```
read_oa
[-help]
oa_lib
oa_cell
oa_view
[-filter {block_insts blockages boundary fixed_core_insts floorplan pad_insts pin_shapes regions regular_routing
special_routing bump_insts}]
[-nets { netnames }]
```

Restores floorplan, placement, and routing data using the OpenAccess format. This command only reads physical information from the OpenAccess database. For information on restoring an OpenAccess database, see read_db.

You can use the `read_oa` command after importing the design.

Note: Innovus supports only the following begin/end extension values for DEF `NETS` style wiring segments: 0, width/2 or LEF `LAYER WIREEXTENSION`. If any of the wire extensions has unsupported extension value, then Innovus converts it to a special wire (DEF `SPECIALNETS`). While writing back OpenAccess (`write_oa` or `write_db -oa_lib_cell_view`), these special wires will again be converted back to DEF `NETS` style wiring.

Parameters

`-help`	Outputs a brief description that includes type and default information for each parameter of the `read_oa` command. For a detailed description of the command and all of its parameters, use the `man` command: `man read_oa`.
oa_lib	Specifies the OpenAccess library name. *Data_type*: string, required
oa_cell	Specifies the name of the top cell to import. *Data_type*: string, required
oa_view	Specifies cellview name to import. *Data_type*: string, required
`-filter {block_insts blockages boundary fixed_core_insts floorplan pad_insts pin_shapes regions regular_routing special_routing bump_insts}`	

	Lists the OpenAccess cellview information data to be read. When not specified, all information is read from the OpenAccess cellview. When `-filter` is specified, only the object types that are chosen will be read. Additionally, when reading pin_shapes or instances, information in the cellview that refers to terminals or instances that do not exist in memory will be ignored. The legal `-filter` options are: • `block_insts` - Indicates that any CLASS BLOCK instances should be updated. Physical-only block instances may be added to the database if the placement status is `fixed` or `cover`. • `blockages` - Indicates that blockages should be processed. If a blockage is attached to an instance that does not exist in the in-memory database, then it is ignored. • `boundary` - Indicates that the design boundary information, including rows and tracks, should be updated. When `boundary` is specified, the design boundary, rows and tracks are created from the new boundary. • `fixed_core_insts` - Indicates that any CLASS CORE instances that have placement status `fixed` or `cover` should be updated. Physical-only core instances may be added to the database if the placement status is `fixed` or `cover`. • `floorplan` - Is equivalent to specifying "`block_insts blockages boundary fixed_core_insts pad_insts pin_shapes regions special_routing`" • `pad_insts` - Indicates that any CLASS PAD instances should be updated. Physical-only pad instances may be added to the database if the placement status is `fixed` or `cover`. • `pin_shapes` - Indicates that pin shapes should be read. When `pin_shapes` is specified, the existing pins are replaced by the ones being read from the cellview. • `regions` - Indicates regions should be updated. • `regular_routing` - Indicates that nets should be processed and regular routing (and associated shield net wiring) and net constraints updated. • `special_routing` - Indicates that nets should be processed and special routing (except shield nets wiring) should be read. • `bump_insts` - Indicates that bump instances from the cell view need to be read. Use `read_oa -filter` to read floorplanning information from a cellview whose connectivity does not match the current Innovus in-memory connectivity. The `-filter` option allows you to load some data selectively, while not loading others. When `-filter` is used, error checking for database consistency (nets, instances, terminals, etc.) is disabled. While similar to Virtuoso Load Physical View (LPV), the `-filter` option is only a subset and not intended to be a complete replacement. *Data_type*: (enum)+, optional
`-nets {`*netnames*`}`	
	Specifies the list of nets whose routing (routing and special) and constraints have to be read with the `-filter` option. Shielding wiring/vias are also read in if they are used to shield any of the specified nets. If this option is used, typically `regular_routing` is not specified as part of the `-filter` objects. *Data_type*: string, optional

Example

- The following command restores data using OpenAccess:
  ```
  read_oa design amba_usb layout
  ```

Related Information

- `read_db`
- `oa_read_write` Category Attributes
- `write_db`
- `write_oa`
- *Mixed Signal (MS) Interoperability Guide*
  - Design Data Preparation chapter
  - Schematic-Driven Mixed Signal Design Flow chapter
  - Virtuoso Digital Implementation chapter
  - Chip Finishing and ECO Flows chapter

register_trigger

```
register_trigger
[-help]
[{-pre | -post}
command_name
proc_name]
```

Registers the `-pre` or `-post` callback procedures with the specified commands.

The procedure registered with `-pre` parameter runs before the *`command_name`* parameter is used while the procedure registered with `-post` parameter runs after the *`command_name`* parameter is used.

Note: The arguments passed to *`command_name`* are passed to *`proc_name`*. If the procedure registered with the `-pre` parameter returns a `TCL_ERROR`, then *`command_name`* is not run.

You can use the `register_trigger` command after starting the Innovus software.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `register_trigger` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man register_trigger`.
`command_name`	Specifies the command name. The supported command names are `read_db`, `write_db`, `read_oa`, and `write_oa`. *Data_type*: string, required
`proc_name`	Specifies the procedure name that is to be registered as callback. *Data_type*: string, required
`-post`	Specifies the procedure to be called after the command. *Data_type*: bool, required
`-pre`	Specifies the procedure to be called before the command. *Data_type*: bool, required

Related Information

- `read_db`
- `read_oa`
- `write_db`
- `write_oa`

report_oa_lib

```
report_oa_lib
[-help]
lib_name
[-header]
[-filter {cdsinfo | compression | constraint_groups | property | tech_graph}]
```

Reports information about OpenAccess libraries from within Innovus. You can use this command to query for attributes that are controlled by some of the oa_read_write Category Attributes and the `update_oa_lib` command.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `report_oa_lib` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man report_oa_lib`.
`lib_name`	Specifies the name of the OpenAccess library to be queried. *Data_type*: string, required
`-filter {cdsinfo | compression | constraint_groups | property | tech_graph}`	
	Specifies the list of information to be included in the report. You can choose from the following filter options: • `cdsinfo` - Reports whether the `cdsinfo.tag` exists in the specified library. • `compression` - Reports the compression level (0 = uncompressed) of the library. • `constraint_groups` - Reports the constraint groups defined in the libraries in the technology graph. Constraint groups that are compatible with the default rule and `NONDEFAULTRULE` requirements will be indicated. • `property` - Reports the property information that was added to the library. The `add_oa_library_property` command is one source of library properties. • `tech_graph` - Reports information about the technology graph from the specified library. If you do not specify a filter option, all the above information is included in the report. *Data_type*: (enum)+, optional
`-header`	Indicates that header information should be displayed. This option is useful if the report is being redirected to a file. *Data_type*: bool, optional

Example

The following command reports information about the `zambezi45` OpenAccess library along with the header:

```
innovus 18> report_oa_lib zambezi45 -header

;*******************************
; Header Info
;*******************************
; Generated by: Cadence Innovus 18.10-a025_1
; OS: Linux x86_64(Host ID donut)
; Generated on: Mon Feb 12 15:53:07 2018
```

```
; Command: report_oa_lib zambezi45


Library Name : zambezi45

cdsinto.tag exists : Yes

Compression Level : 0

Rapid PDK : No

Library Properties :
#prop name prop type prop value
cdfData AppProp (promptWidth 175 fieldHeight 35 fieldWidth 350 buttonFieldWidth 340 formInitProc nil doneProc nil
parameters nil propList (instDisplayMode "instName" instNameType "schematic" termDisplayMode "none" termSimType
"DC" netNameType "schematic" paramDisplayMode "parameter" paramSimType "DC" paramEvaluate "t nil nil nil nil"
simInfo (nil)))
techLibName StringProp gsclib045

Constraint Groups : 10
Library Name: gsclib045
LEFDefaultRouteSpec (LEFDefaultRouteSpec, default)
validLayers: Metal1 Metal2 Metal3 Metal4 Metal5 Metal6 Metal7 Metal8 Metal9 Metal10 Metal11
validVias:

LEFSpecialRouteSpec
Library Name: gpdk045
LEFDefaultRouteSpec_gpdk045 (LEFDefaultRouteSpec, default)
validLayers: Metal1 Metal2 Metal3 Metal4 Metal5 Metal6 Metal7 Metal8 Metal9 Metal10 Metal11
validVias:

virtuosoDefaultExtractorSetup
virtuosoDefaultSetup
virtuosoDefaultTaper
VLMDefaultSetup
DFM
minPRBoundaryInteriorHaloCG
minProtrusionNumCutCG

Technology Graph:
zambezi45
gsclib045 (attached)
gpdk045 (referenced)
```

Related Information

- `oa_read_write` Category Attributes
- `update_oa_lib`
- *Mixed Signal (MS) Interoperability Guide*
  - Technology Data Preparation chapter
  - Static Timing Analysis for Mixed Signal Designs chapter

set_oa_cell_binding

```
set_oa_cell_binding
[-help]
[-abstract ]
[-oa_lib list_of_libs]
{-base_cell list_of_cell_patterns -oa_view list_of_views}
```

Changes the layout binding for a cell. This also affects the layout viewing and automatic remastering. When specified with `-abstract` parameter, this instead of affecting the abstract binding replaces the in-memory cell information with the matched cells.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `set_oa_cell_binding` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man set_oa_cell_binding`.
`-abstract`	Replaces in-memory cell information for the matched cells. *Data_type*: bool, optional
`-base_cell list_of_cell_patterns`	Specifies one or more cell name patterns that are to be replaced. *Data_type*: string, required
`-oa_lib list_of_libs`	Specifies the OpenAccess library that is to be used to look for cell abstracts. *Data_type*: string, required
`-oa_view list_of_views`	Specifies the cellview name. *Data_type*: string, required

Examples

- The following command looks for the cells in Innovus database that match the cell pattern and replaces their binding with the specified view found in the library list.

```
set lib_list1 {lib1 lib2}
set_oa_cell_binding -oa_lib $lib_list1 -base_cell {cell1 cell2 cell3} -oa_view {abc}
```

Related Information

- `write_oa`

unlock_oa_design

```
unlock_oa_design
[-help]
oa_lib
oa_cell
oa_view
```

Unlocks the specified OpenAccess design.

A design cannot be opened if `set_db oa_lock` is set to `1`, and the design is currently locked. You can use the `unlock_oa_design` command to free the lock so that other tools can access and edit the database. You also can use the command to override the lock, if the design is incorrectly locked.

You can use the `unlock_oa_design` command at any time.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `unlock_oa_design` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man unlock_oa_design`.
`oa_lib`	Specifies the name of the OpenAccess oa_library in which the design was saved. *Data_type*: string, required
`oa_cell`	Specifies the name of the top oa_cell to unlock. *Data_type*: string, required
`oa_view`	Specifies the name of the oa_cell oa_view to unlock. *Data_type*: string, required

Example

- The following command unlocks the oa_view `layout` in the oa_cell `dtmf_chip` in the oa_library `designLib`:
  ```
  unlock_oa_design designLib dtmf_chip layout
  ```

Related Information

- oa_read_write Category Attributes
  - `oa_lock`

update_oa_lib

```
update_oa_lib
[-help]
lib_name
[-compress_level value]
[-create_cdsinfo_tag]
[-tech_attach_to_reference]
```

Updates the specified OpenAccess library from within Innovus. You can use this command to:

- Converts the specified library to use a tech reference from the attached tech so that the non-parameterized or custom vias can be written to it. If an OpenAccess design has non-parameterized vias, you should run `update_oa_lib` before invoking `write_db`. The tool is then able to save the design and does not throw an error because it can now write non-parameterized vias to the converted library.
- Add a cdsinifo.tag to legacy libraries created by earlier versions of Innovus.
- Modify the compression level for the library.

If you run the `update_oa_lib` command on a library that is not in attached mode, a warning is displayed. However, the script continues to run. This is useful if you want to make sure that libraries are writable for new vias and the `write_db` command does not fail.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `update_oa_lib` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man update_oa_lib`.
`-compress_level` *value*	Sets the compression level of the specified library. Cellviews written to the library follow the specified compression level. However, existing cellviews in the library are not modified. Use the oazip utility to modify the existing cellviews in the library. Valid compression level values range from 0 to 9. A value of 0 indicates that compression is turned off. *Data_type*: int, optional
`-create_cdsinfo_tag`	Adds a `cdsinfo.tag` file to the specified library. *Data_type*: bool, optional
lib_name	Specifies the name of the library to be updated. *Data_type*: string, required
`-tech_attach_to_reference`	Converts the specified library to use a tech reference from the attached tech so that new custom via definitions can be added to the library. *Data_type*: bool, optional

Examples

- The following command sets the compression level of the `lib1` library to 9. However, it does not change the compression of the existing cellviews in the library:
  ```
  update_oa_lib lib1 -compress_level 9
  ```
  The tool displays warning such as the following:
  ```
  Changed library compressLevel from '0 (uncompressed)' to '9'. Cellviews within the library are not affected by changing the library's compressLevel. To update the cellviews, use the oazip utility.
  ```
- The following command decompresses the `my_lib` library, which previously is at compression level `1`:
  ```
  update_oa_lib my_lib -compress_level 0
  ```

Related Information

- `oa_read_write` Category Attributes
- *Mixed Signal (MS) Interoperability Guide*
  - Static Timing Analysis for Mixed Signal Designs chapter

write_lef_library

```
write_lef_library
[-help]
[lef_file]
[-all_auto_generated_via | -macro_only]
[-macro_only [-no_property]]
[-tech_only | -macro_only]
[-tech_only | -macro_list string]
```

Writes the LEF library data (composed of technology data and macros) that is present in the Innovus database.

The `write_lef_library` command facilitates comparisons between LEF-based input and OpenAccess-based input. LEF allows cases where `VIARULE GENERATE` has `WIDTH` specified for either the top or bottom layer. However, such cases are not supported by the OpenAccess tech. To make the comparison between LEF-based and OpenAccess-based input easier, `write_lef_library` dumps `"WIDTH 0 TO 1000 ;"` for any `VIARULE GENERATE` that does not have WIDTH specified in the original LEF.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `write_lef_library` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man write_lef_library`.
`lef_file`	Specifies the output LEF file name. *Data_type*: string, optional
`-all_auto_generated_via`	
	Includes all auto-generated vias from the `add_route_via_defs` command. *Data_type*: bool, optional
`-macro_list` *`string`*	
	Writes the macros specified in the macro list. *Data_type*: string, optional
`-macro_only`	Writes only the macro data to the LEF file. *Data_type*: bool, optional
`-no_property`	Writes data in the LEF file without any properties. This option can be used only with the `-macro_only` option. *Data_type*: bool, optional
`-tech_only`	Writes only the technology data to the LEF file. *Data_type*: bool, optional

Examples

- The following command writes the technology data into the `tech.lef` file:
  ```
  write_lef_library tech.lef -tech_only
  ```
- You can set the LEF version to 5.7 or 5.8 using `write_def_lef_out_version`:
  ```
  set_db write_def_lef_out_version 5.8
  ```

Related Information

- `add_route_via_defs`
- `write_oa_techfile`
- *Mixed Signal (MS) Interoperability Guide*
  - Technology Data Preparation chapter
  - Design Data Preparation chapter
  - Working with Vias in OpenAccess chapter

write_oa

```
write_oa
[-help]
oa_lib
oa_cell
oa_view
[-auto_remaster]
[-cut_rows]
[-early_global_route]
[-no_overwrite_abstract]
[-oa_leaf_views { lefView1 lefView2 ... }]
[-oa_ref_libs { refLib1 refLib2 ... }]
[-no_connectivity [-no_std_cells]]
```

Saves floorplan, placement, and routing data in the OpenAccess format.

You can use the `write_oa` command after importing the design.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `write_oa` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man write_oa`.
oa_lib	Specifies the target OpenAccess library name. *Data_type*: string, required
oa_cell	Specifies the target cell name. *Data_type*: string, required
oa_view	Specifies the target view name. *Data_type*: string, required
`-auto_remaster`	Overrides the reference libraries and leaf view names specified binding, and utilizes the cell layout viewing binding stored on the library cell master for each leaf cell (stored in the library cell's layout library and layout view attributes). The binding can be specified using the `set_oa_cell_binding` command. *Data_type*: bool, optional
`-cut_rows`	Specifies that rows are cut against placement obstructions and block halos, similar to the behavior of the `write_def` command. *Default:* Does not cut rows. *Data_type*: bool, optional
`-early_global_route`	Writes out a design containing Early Global Route data. *Data_type*: bool, optional

`-no_connectivity`	Writes out a design that contains instances and physical shapes. When specified, the software creates an OpenAccess module domain using the instance hierarchy, but does not attempt to create any nets, pins, or instance pins. This behavior is similar to writing a DEF file without the NETS section. **Note** - Designs written out using the `-no_connectivity` parameter cannot be reloaded into the Innovus software. *Data_type*: bool, optional
`-no_overwrite_abstract`	Prevents the software from overwriting abstracts of partitions and blackboxes. *Default:* Innovus overwrites abstracts of partitions and blackboxes. Innovus never overwrites abstracts of library data. *Data_type*: bool, optional
`-no_std_cells`	Does not save instances of core cells to the OpenAccess database. However, if an instance is either fixed or covered, it is saved to the database. Use this parameter if you just need the preliminary design for DRC checks of the floorplan. In such cases, leaving out the standard cell information makes the file smaller and reduces the DRC run time. **Note** - The `-no_std_cells` parameter must be used with the `-no_connectivity` parameter. *Data_type*: bool, optional
`-oa_leaf_views {lefView1 lefView2 ...}`	
	Specifies a list of view names to be searched during leaf cell instance binding. *Data_type*: string, optional
`-oa_ref_libs {refLib1 refLib2 ...}`	
	Specifies the names of the external libraries that will contain the leaf cell masters, LEF abstract, and technology file. This parameter only accepts library names, not paths. The mapping between the external library name and its path is defined in the `cds.lib` file. Innovus references technology information only from the first library in the list of libraries. *Data_type*: string, optional

Example

- The following command shows the usage of ITDB style libraries:

```
create_oa_lib myDesignLib -oa_reference_tech_libs {techLib}
write_oa -auto_remaster myDesignLib myTopCell layout
```

Related Information

- `create_oa_lib`
- `read_oa`
- `set_oa_cell_binding`
- `write_def`
- *Mixed Signal (MS) Interoperability Guide*
  - Design Data Preparation chapter

write_oa_abstract

```
write_oa_abstract
[-help]
[-cut_obs]
[-cut_obs_min_spacing]
[-extract_block_obs]
[-extract_block_pg_pin_layers layerNameList]
[-oa_lib lib]
[-oa_view view]
[-pg_pin_layers layerNameList]
[-stripe_pin {top all none}]
[-top_layer topLayerName]
```

Generates hierarchical design abstract information for the current block-level design. By default, if `write_oa_abstract` creates power and ground pins on the design and creates a single obstruction shape for each routing layer. The output abstract contains the antenna information, if you run antenna verification.

In a bottom-up flow (where the block was not created using the `create_partition` or `commit_partitions` command, you normally create a simple OpenAccess abstract that contains:

- one obstruction (`OBS`) shape per routing layer
- signal pins located along the block boundary
- power pins for the top metal layer

You can use the `write_oa_abstract` command after loading a block-level design.

Parameters

`-help`	Prints a brief description that includes the type and default information for each `write_oa_abstract` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man write_oa_abstract`.
`-cut_obs`	Creates cut-outs (obstructions) in the blockages for signal pins, power and ground pins, vias, and power and ground stripes. *Default*: Creates a single obstruction shape for each routing layer. *Data_type*: bool, optional
`-cut_obs_min_spacing`	Creates cut-outs (obstructions) in the blockages around pins using the minimum width to calculate the spacing. *Data_type*: bool, optional
`-extract_block_obs`	Extracts all obstruction (`OBS`) information from the instantiated `CLASS BLOCK` or `RING` macros and writes out the obstructions to the output OpenAccess file. `OBS` information is extracted for layers that are not already being completely covered under the normal routing layer range and for cut layers between routing layers that are completely covered. *Data_type*: bool, optional
`-extract_block_pg_pin_layers layerNameList`	

	Extracts the power and ground pin information from any CLASS BLOCK and CLASS RING macros with pins on layers in the specified layer list and writes out the pin information to the OpenAccess file. You can use metal layer numbers (for example, specify `4` for `metal4)`. Specify a list of layer numbers by leaving a space between each number. For example, to create a list with layers `metal3`, `metal4`, and `metal5`, specify `-extract_block_pg_pin_layers 3 4 5`. *Default*: Does not extract the pin information from any macros. *Data_type*: string, optional
`-oa_lib` *lib*	Specifies the target library name. *Default*: The abstract is written into the library from which the current data was loaded. *Data_type*: string, optional
`-oa_view` *view*	Specifies the target view name. *Default*: `abstract` *Data_type*: string, optional
`-pg_pin_layers` *layerNameList*	
	Specifies the list of layers. *Data_type*: string, optional
`-stripe_pin {top all none}`	Writes out power and ground stripes on the top metal layer as power and ground pins. You can use this parameter in a bottom-up methodology, when you create abstracts for blocks that you will use at the next higher level of a hierarchical design. *Data_type*: enum, optional
`-top_layer` *topLayerName*	Writes out obstruction information only for layers up to the specified layer. You can use the metal layer number (for example, specify `4` for `metal4`). *Default:* Top routing layer (therefore, writes out obstruction information for all layers) *Data_type*: string, optional

Related Information

- `commit_partitions`
- `create_partition`
- `write_lef_abstract`
- *Mixed Signal (MS) Interoperability Guide*
  - Useful Tips chapter

write_oa_black_boxes

```
write_oa_black_boxes
[-help]
-base_cells list_of_cells
-oa_lib libName
[-oa_view viewName]
```

Saves the specified blackbox abstracts, using the OpenAccess database format. You can use the `write_oa_black_boxes` command at any time after importing the design.

Note that when the design is saved using the `write_db` command, the tool automatically saves abstracts for any blackboxes in the design. However, if you are performing pin optimization or manipulating the blockages on a particular macro in the design, you may wish to save the abstract for only that particular macro. The `write_oa_black_boxes` command is useful in such situations.

Parameters

`-help`	Prints a brief description that includes type and default information for each `write_oa_black_boxes` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man write_oa_black_boxes`
`-base_cells` *`list_of_cells`*	Specifies one or more blackboxes to save. If you specify more than one blackbox, you must enclose them in either double quotation marks or curly braces, and separate them with an empty space. *Data_type*: string, required
`-oa_lib` *`libName`*	Specifies the OpenAccess library to use to save the blackbox abstracts. If an OpenAccess library with the same name already exists, the software creates a cell view in that library. If the library does not exist, the software creates the library in the working directory, with the library name and the path name being the same. If a new library is created, the technology file is copied from the technology library to the new library. *Data_type*: string, required
`-oa_view` *`viewName`*	Specifies the cell view to save. *Default:* Uses the view name `abstract` *Data_type*: string, optional

Examples

- The following command saves the blackbox `myBlackbox` in view `abstract`:
  ```
  write_oa_black_boxes -oa_lib myLib -base_cells myBlackbox -oa_view abstract
  ```
- The following command saves the blackboxes `myBlackbox1` and `myBlackbox2` in view `abstract`:
  ```
  write_oa_black_boxes -oa_lib myLib -base_cells "myBlackbox1 myBlackbox2" -oa_view abstract
  ```
- The following command also saves the blackboxes `myBlackbox1` and `myBlackbox2` in view `abstract`:
  ```
  write_oa_black_boxes -oa_lib myLib -base_cells {myBlackbox1 myBlackbox2} -oa_view abstract
  ```

Related Information

- write_db

write_oa_techfile

```
write_oa_techfile
[-help]
[out_file]
[-include_rules]
[-process_node coord]
```

Writes out a DFII style ASCII technology file that can be given to the `techLoadDump` executable to create an equivalent OpenAccess technology database.

This command is used to create and verify OpenAccess technology files that match LEF technology files. By default, `write_oa_techfile` writes out only DRC information and no LDRS or VIA information.

The expected flow for new processes is as follows:

1. Read a LEF file into Innovus with `init_design`.
2. Use `write_oa_techfile` to write out the tech (`.tf`) file.
3. Run `techLoadDump` using the `.tf` file.
4. Use `lef2oa -pnrLibDataOnly` to create the LDRS and vias.

For more information, refer to the chapter "Technology File Layer Attributes" of the *Virtuoso Technology Data ASCII Files Reference*.

Parameter

`-help`	Outputs a brief description that includes type and default information for each `write_oa_techfile` parameter. For a detailed description of the command and all of its parameters, use the man command: man `write_oa_techfile`.
out_file	Specifies the output ASCII file name. *Data_type*: string, optional
`-include_rules`	Includes LDRS and VIA information in the output tech file. This option should be used only if the LEF default and NDRs do not utilize fixed vias. *Data_type*: bool, optional
`-process_node` *coord*	Specifies a process node value in nm. When specified, `write_oa_techfile` creates a processNode entry in the techfile for processes using the MASK construct. For example, if you specify `-process_node 20` to indicate a 20nm process, the following entry is included in the control() section of the techfile: `processNode(0.020)` If `-process_node` is not specified and the `write_oa_techfile` command detects MASK constructs on any layer, it returns an error. *Data_type*: int, optional

Examples

- The following set of commands updates an OpenAccess technology file with any missing information from a LEF technology file:

```
#Initialize a design using LEF files
write_lef_library -tech_only lef.lef  #A LEF file in canonical
                                      #sorted order for easy
                                      #comparison with diff
write_oa_techfile lef.tf              #A DFII ASCII tech file
                                      #that can be given to
                                      #techLoadDump executable
#Edit existing OA tech file to add any missing information in
#the lef.tf file
#Initialize a design using the new OA tech. Then:
write_lef_library -tech_only oa.lef #A LEF file in canonical
                                    #order derived from OA tech
#Compare the two LEF files. Any differences need to be fixed
#or explained.
diff lef.lef oa.lef
```

Related Information

- write_lef_library
- *Mixed Signal (MS) Interoperability Guide*
  - Technology Data Preparation chapter
  - Routing Constraint Interoperability chapter

24

Partition Commands and Attributes

- add_connection_to_neighbor_partition
- add_partition_feedthrus
- align_partition_clones
- align_pins
- assemble_design
- assign_io_pins
- assign_partition_pins
- assign_pins Category Attributes
- change_current_design
- check_bus_guides
- check_hier_route
- check_pin_assignment
- clone_partition_obj
- commit_partitions
- commit_pushdown_eco
- compare_pin_assignment_statistics
- connect_macro_feedthru
- convert_blackbox_to_fence
- convert_fence_to_black_box
- create_blackbox
- create_net_group
- create_partition
- create_partition_feedthru
- create_partition_wrapper
- create_pin_blockage
- create_pin_group
- create_pin_guide
- create_pushdown_eco
- delete_all_partition_cuts
- delete_all_partition_feedthrus
- delete_black_boxes
- delete_net_group
- delete_nets_from_net_group
- delete_partition_cuts
- delete_partitions
- delete_pin_blockages
- delete_pin_from_pin_group
- delete_pin_groups
- delete_pin_guides

- edit_pin
- fix_feedthru_assign
- flatten_cover_cell
- flatten_partitions
- get_black_box_area
- get_clone_partition_orient
- get_partition_pin_status
- get_pin_constraint
- gui_highlight_feedthru_nets
- gui_show_partition_wire_cross_over
- highlight_sync_obj
- legalize_pins
- partition_floorplan Category Attributes
- partition_push_network
- partition_push_special_wires
- place_cells_at_center_for_feedthrus
- place_partition_clone
- read_black_box_netlist
- read_hierarchical_def
- read_partition_pins
- report_partitions
- report_pin_assignment_statistics
- report_two_pin_chain
- report_unaligned_nets
- reset_partition_pins
- reset_pin_constraint
- route_hierarchical_clock
- set_blackbox_master_orient
- set_blackbox_master_to_r0
- set_clone_partition_orient
- set_partition_area_io
- set_partition_pin_status
- set_partition_pin_use
- set_pin_constraint
- set_promoted_macro_pin
- undo_uniquify_partition
- uniquify_partition
- update_net_group
- update_partition_cell_obs
- update_partition_cut
- update_pin_group
- write_black_box_lef
- write_generalized_feedthru_paths

- write_net_groups_topo
- write_partition_pins
- write_partitions

add_connection_to_neighbor_partition

```
add_connection_to_neighbor_partition
[-help]
```

Adds a partition pin on the abutted edge of unique abutting partitions to create a 2-partition pin scenario for target pins. This helps to avoid floating pins on abutted edges. The `add_connection_to_neighbor_partition` command targets the following pins:

- Input partition pins with have tie definition and a GNC/UPF to resolve the tie net to a supply
- Floating output partition pins

Note: The target pins may or may not have any load/driver inside the partition.

This command should be used after the pins have been assigned for the target partition and tie cells have been specified using the `add_tieoffs_cells` attribute.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `add_connection_to_neighbor_partition` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man add_connection_to_neighbor_partition`

Example

- The following command adds partition pins on abutting partitions and reports the number of new partition pins and tie cells in a table.

```
add_connection_to_neighbor_partition
-------------------------------------------------------------------------------------
|    Type     | Total Pins | Pins placed in abutted edge | New partition pins added |
-------------------------------------------------------------------------------------
|   Tie-low   |          4 |                           4 |                        4 |
|   Tie-hi    |          4 |                           4 |                        4 |
| Floating(I) |         16 |                          16 |                        - |
| Floating(O) |         16 |                          16 |                       16 |
-------------------------------------------------------------------------------------
```

Related Information

- Partitioning the Design chapter in the *Innovus Stylus Common UI User Guide*

add_partition_feedthrus

```
add_partition_feedthrus
[-help]
[-all_lower_case]
[-blockage_file file_name]
[-blocked_edges_file file_name]
[-inst_prefix instance_prefix]
[-net_prefix net_prefix]
[-reuse_buffer {true false 0 1}]
[-symmetrical_repeat_abutted_floorplan]
[-use_short_name]
[-verbose]
[-write_topological_file topology_file_name]
[[[-include_nets_file file_name | -selected_nets | -auto_select_nets_by_congestion | -auto_select_nets_by_slack |
[ [-exclude_nets_file file_name] ]] ] | -use_topological_file topology_file_name ]
[ -use_topological_file topology_file_name | -route_based | -route_based_strict |
[-exclude_partitions {partition_name | partition_name_list} ] ] [ -use_topological_file topology_file_name | -route_based | -
route_based_strict ]
[ -route_based [-auto_select_nets_by_slack ] [-auto_select_nets_by_congestion ]]
[-no_buffer | [[-buffer_cell {cell_name | cell_name_list}] [-double_buffer ]]]
[[-check_only ] | [[-eco_file file_name] [-net_mapping file_name]] ] [-use_topological_file topology_file_name ] [-no_buffer ]
```

Inserts feedthrough buffers into the partitions, changing the original netlist, to avoid routing a net over a block area. ECO placement runs automatically with this command. The buffer supplied during design import is taken as default. For Alternately, you can specify a buffer with the `-buffer_cell` parameter or specify the `-no_buffer` parameter to generate an ASSIGN statement in the netlist. You can use this command after the floorplanning steps for a partitioned design.

The `add_partition_feedthrus` command can detect if the design has power domains. This way, appropriate buffers which are inserted into power domains will automatically be derived based on power domain library binding. An error message is reported if no regular buffer is found for an *Always On* power domain in the feedthrough path. Both placement and route based feedthrough insertions can support *Always On* and switchable power domains.

You can save the feedthrough buffer topology tree information in a file by using the `-write_topological_file` parameter. You can later use the `-use_topological_file` parameter to replicate feedthrough buffer insertions for an ECO netlist by specifying the file that has the topology tree information. You can also manually create a topology file to guide feedthrough insertion for specific nets.

Notes:

- The `add_partition_feedthrus` command by default derives the feedthrough topology using a shortest-path algorithm, which is based on the placement results.
- The `add_partition_feedthrus` command by default considers a design to be channel-less, having no channel available for routing. Therefore all nets are candidates for feedthrough insertion.
  The default behavior assumes a true channel-less design - that is, the possibility of routing through channels is considered minimal. It does not consider routing congestion, routing blockages, route guides, pin guides, and bus guides. If these considerations are required, use the `-route_based` parameter to derive the feedthrough topology based on the routing results.
- Feedthroughs are inserted in nets that connect to floating ports. Floating ports are hierarchical ports that are not connected to any logic inside the module.
- The `add_partition_feedthrus` command removes nets that are inserted with feedthrough buffers from any net groups to which they belong. After running this command you should, therefore, update the net groups that contain feedthrough nets.
- If a pin guide exists for a net for which feedthrough buffers are inserted, the pin guide information is updated.
- The `add_partition_feedthrus` command will handle designs with master/clone partitions automatically. However, it may create infeasible pin solution for channel-less designs with fully abutted partitions.
- The `add_partition_feedthrus` command supports nested partitions. For inserting feedthrough buffers into nested partitions, the `add_partition_feedthrus` command, first, automatically deletes the parent partitions and keeps only the bottom-most partition. It then does a single-level feedthrough insertion and then automatically restores the parent partitions and brings the nested partition definition back as it is. In nested designs, the `add_partition_feedthrus` command only puts buffers and ports in the bottom most level of partitions.
- The `add_partition_feedthrus` command prints a summary of all messages issued for nets in the log file.
- A feedthrough always uses a low power compatible buffer/AO-buffer. If it cannot use a low power complaint cell, then the default buffer (specified with the `-buffer_cell` parameter) is used, and the buffer name is prefixed with “MSV_INVALID”.

Parameters

`-all_lower_case`	Specifies all lowercase characters for the inserted feedthrough buffer and feedthrough net names. *Data_type*: bool, optional
`-auto_select_nets_by_congestion`	Specifies that the critical nets in congested channels will be identified automatically. These critical nets are routed flat so that later they can be feedthrough over partition(s) to reduce congestion, while other remaining nets are routed by honoring fences. Once the routing is done, feedthroughs are inserted based on the routing result. *Data_type*: bool, optional
`-auto_select_nets_by_slack`	Specifies that the critical nets in the design will be identified automatically. These critical nets are routed flat so that later on they can be feedthrough over partition(s) to improve timing while other remaining nets are routed by honoring fences. Once the routing is done, feedthroughs are inserted based on the routing result. *Data_type*: bool, optional
`-blockage_file` *file_name*	Specifies a file that contains the coordinates of line segments that are considered as blockages for placement based feedthrough insertion. The following is the format of the blockage file: *x1 y1 x2 y2* *x1 y1 x2 y2* *...* *xn yn xm ym* The blockage lines are specified as coordinates of a box, however, the boxes should be of zero width or height. *Data_type*: string, optional
`-blocked_edges_file` *file_name*	Specifies a file that contains the names of the partitions followed by the edge numbers for the partition which are considered as blocked for the purpose of `add_partition_feedthrus` command. *Data_type*: string, optional
`-buffer_cell {`*cell_name* `|` *cell_namel_list*`}`	
	Specifies the name of the feedthrough buffer cell type. **Note:** You can specify multiple buffer cells with MSV designs. The first buffer will be used for the default power domain. *Data_type*: string, optional
`-check_only`	Specifies that the feedthrough buffers are *not* inserted. This parameter is useful when you want to know how the `add_partition_feedthrus` command will affect the design. A summary information will be printed. You can specify this parameter along with the `-write_topological_file` parameter. This will save the feedthrough topology information. *Data_type*: bool, optional
`-double_buffer`	Adds a buffer close to the feedthrough input pin and adds a buffer close to the feedthrough output pin. *Data_type*: bool, optional
`-eco_file` *file_name*	Specifies the file for logging ECO directives. This parameter cannot be used with the `-check_only` parameter. *Data_type*: string, optional
`-exclude_nets_file` *file_name*	
	Specifies a file that contains net names to be excluded from being a feedthrough net for a partition. *Data_type*: string, optional
`-exclude_partitions {`*partition_name* `|` *partition_list*`}`	
	Specifies that feedthrough buffers should not be added to the specified partitions. *Data_type*: string, optional
`-help`	Outputs a brief description that includes the type and default information for each `add_partition_feedthrus` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man add_partition_feedthrus`
`-include_nets_file` *file_name*	Specifies a file that contains net names to be feedthrough nets for a partition and inserted feedthrough buffers. *Data_type*: string, optional

`-inst_prefix` *instance_prefix*	Specifies a prefix for the newly created instances. The prefix that you specify will be used instead of the default prefix. *Data_type*: string, optional
`-net_mapping` *file_name*	Generates a file that maps the original net name to the new net name created for the feedthrough buffers. **Note:** You must specify this parameter if you later want to run the `gui_highlight_feedthru_nets` command to highlight the feedthrough paths of the nets. *Data_type*: string, optional
`-net_prefix` *net_prefix*	Specifies a prefix for the newly created nets. The prefix that you specify will be used instead of the default prefix. *Data_type*: string, optional
`-no_buffer`	Generates an `ASSIGN` statement in the netlist instead of inserting a buffer. *Data_type*: bool, optional
`-reuse_buffer {true | false | 0 | 1}`	
	Reuses inserted buffers across multiple instantiation of a module within one run. This option should only be used for designs that have master and clone partitions. *Default*: `true` *Data_type*: enum, optional
`-route_based`	Derives the feedthrough topology based on the routing results. *Data_type*: bool, optional
`-route_based_strict`	Derives the feedthrough topology based on the routing results and follows each criss-cross. In comparison to the `-route_based` parameter, the `-route_based_strict` option inserts more number of buffers depending on the routing cross-overs and also supports special routes. It can insert buffers on intra-partition net, if the routing crosses outside the boundary. However, such a net needs to be handled via selected feedthrough insertion and not default selection. The `-route_based_strict` parameter places the pins at each of the routing cross-point. **Note:** You can use the `gui_show_partition_wire_cross_over` command to get `[design_name].nets_not_crossing_mc.txt` for a selection of nets on which to use this option. *Data_type*: bool, optional
`-selected_nets`	Specifies that feedthrough buffers should be inserted on the nets that are currently selected in the GUI. *Data_type*: bool, optional
`-symmetrical_repeat_abutted_floorplan`	
	This option is relevant to master clone designs. It is to be used for the floorplans which have repeated components arranged in a symmetric fashion with a data flow going from the top to each component independently. If the design has nested partitions, it will delete all the partitions which are parents. It will hold more partitions within them, perform its operation and restore the parent partitions once finished. *Data_type*: bool, optional
`-write_topological_file` *file_name*	
	Saves a file with feedthrough buffer topology tree information from an existing floorplan. *Data_type*: string, optional
`-use_topological_file` *topology_file_name*	

	Specifies the file that has the topology tree information. This information is used to create feedthrough buffers for the netlist. If you use the `-use_topological_file` parameter, only those nets that are specified in the topology file are considered for feedthrough buffer insertion. If a net does not exist in the design, it should not be in the topology file. For example, if ECO changes remove a net, that net should be removed from the topology file. **Note:** Hierarchical bus names are not supported in the Innovus software. Therefore, you cannot use hierarchical bus names in the topology file. **Note:** The names of the feedthrough nets, buffers, and ports created might change in the different runs of the command. Only the feedthrough path (the partitions through which the net will pass) will always remain the same across the different runs of the command. *Data_type*: string, optional
`-use_short_name`	Specifies the abbreviation of inserted feedthrough net names so that the net names do not extend too long for multiple runs. *Data_type*: bool, optional
`-verbose`	Specifies that net-specific information is included in the logs. *Default:* Only the summary information is included in the logs. *Data_type*: bool, optional

Examples

- The following command generates a file, `K_ABC.topo`, with the feedthrough buffer topology tree information:
  ```
  add_partition_feedthrus -write_topological_file K_ABC.topo -net_mapping abcfile
  ```
  The following is the generated topology file:
  ```
  ############################################################
  #  Generated by: Cadence Innovus
  #  OS:           Linux x86_64(Host ID nofcnl040)
  #  Generated on: Mon Jan 21 12:34:59 2015
  #  Design:       test
  #  Command:      add_partition_feedthrus -write_topological_file K_ABC.topo
  ############################################################
  version 1.0;
  net K_ABC_NET
  hinst-hinst p4 p8;
  hinst-hinst p4 p7;
  end net
  ```
- The following command specifies that topology tree information from the `topology_info` file should be used to generate feedthrough buffer insertions:
  ```
  add_partition_feedthrus -use_topological_file topology_info
  ```

Related Information

- `add_repeater_by_rule`
- `gui_highlight_feedthru_nets`
- `gui_show_partition_wire_cross_over`
- `create_partition_feedthru`
- `delete_all_partition_feedthrus`
- Partitioning the Design chapter in the *Innovus Stylus Common UI User Guide*
  - Inserting Feedthroughs

align_partition_clones

```
align_partition_clones
[-help]
[partition]
[-out_file filename]
[-check_only | -snap_all_corners ]
[ [[-pg_horizontal_grid ] [-pg_vertical_grid ] [-pg_layers {layeridList}] [-update_user_grid ] [-snap_all_corners ] [-skip_finfet_grid ]] |
[[-symmetry_pattern_reference string [-symmetry_pattern_target string
 [[-target_orientation {r0 r90 r180 r270 mx mx90 my my90}][-target_origin {x y}] ]]] | [-symmetric_orientation ]]
 [-pg_layers {layeridList} [-pg_net pgNetName | -pg_center_line float]] [-pg_pitch float [-pg_center_line float]]
```

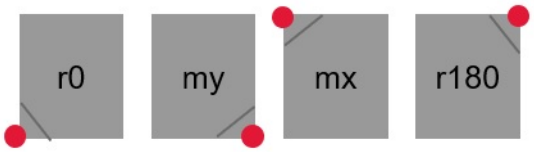
Aligns partition clones with the master partition on a power mesh and with routing tracks. You can use this command after specifying a master partition and clone partition(s). This will avoid off-track pins in clones and avoid potential DRC violations due to misalignment of power meshes inside clone.

Note: The power mesh must be regular for `align_partition_clones` to work correctly.

The `align_partition_clones` command detects and highlights any gap or overlap of fence created in an abutted floorplan.

Parameters

`-check_only`	Specifies that the design should not be modified. When enabled, it only reports messages. *Data_type*: bool, optional
`-help`	Outputs a brief description that includes the type and default information for each `align_partition_clones` param man `align_partition_clones`
`-out_file filename`	Specifies the name of the output file in which to save the master and clone alignment report with details. **Note:** The `align_partition_clones` command supports tracks with multiple definitions. The `-out_file` paramete *Data_type*: string, optional
`-pg_center_line float`	Specifies the centerline of the first virtual symmetric pg stripe (in microns). This option can be repeated. *Data_type*: float, optional
`-pg_layers {layeridList}`	Specifies layer(s) for the pg stripe. This option can be repeated. *Data_type*: string, optional
`-pg_horizontal_grid`	Provide information about horizontal PG grid. *Data_type*: bool, optional
`-pg_net pgNetName`	Specifies name of the pg stripe net name. This option can be repeated. *Data_type*: string, optional
`-pg_pitch float`	Specifies pitch of the pg stripe net (in microns). This option can be repeated. *Data_type*: float, optional
`-pg_vertical_grid`	Provide information about vertical PG grid. *Data_type*: bool, optional
partition	Specifies name of the partition. **Note:** If the partition name is not specified, the command is run on all partitions. *Data_type*: string, optional
`-skip_finfet_grid`	Disables alignment with the FinFet grid. *Data_type*: bool, optional
`-snap_all_corners`	Snaps all corners of master and its clone partition to the user grid. **Note:** The `-snap_all_corners` parameter fixes partition overlaps. *Data_type*: bool, optional

`-symmetric_orientation`	Checks the orientation between each pair of partitions and fixes the symmetry by finding the optimum symmetric c **Note:** If Innovus suggests changing the orientation of a partition, it indicates it as (diff) in the report for quick identi **Note:** The `-symmetric_orientation` parameter can find orientations based upon the floorplan. Even if the `align_p` orientations. It supports nested designs. *Data_type*: bool, optional
`-symmetry_pattern_reference` *string*	Specifies the names of the partitions that are used as a reference set to maintain a symmetrical pattern for the oth **Note:** The first partition in the list is considered as the seed partition from which all the relative locations and orier *Data_type*: string, optional
`-symmetry_pattern_target` *string*	Specifies the names of the partitions that are used as a target set to maintain a symmetrical pattern relative to the **Notes:** • The number of partitions and their order must be the same as that specified with the `-symmetry_pattern_refer` • The current location and orientation of the seed partition is used to relatively move the other partitions in the lis locations and orientations are calculated for the rest of the partitions in the list. • You can specify master partitions in the target partitions set list. However, master partitions can only have r0 or *Data_type*: string, optional
`-target_orientation {r0 r90 r180 r270 mx mx90 my my90}`	
	Specifies the new orientation of the seed partition of the target partition set (`-symmetry_pattern_target`). The spec to maintain the relative orientation of the different pairs of clones. **Notes:** • The seed partition is the first partition specified with the `-symmetry_pattern_target` parameter from which all th • The orientation of the master partitions cannot be changed. *Data_type*: enum, optional
`-target_origin {x y}`	Specifies the new origin of the seed partition of the target partition set (`-symmetry_pattern_target`). The specified  **Notes:** • It does not let the target seed partition go outside the core boundary. • The seed partition is the first partition specified with the `-symmetry_pattern_target` parameter from which all th *Data_type*: point, optional
`-update_user_grid`	Updates the user grid with the new values. *Data_type*: bool, optional

Example

- The following command aligns the partition clones with the master partition and displays a report:
  ```
  align_partition_clones
  ```

  ```
  ================================================================================
                  align_partition_clones Summary
  ================================================================================
  Partition                    | Orientation |   Status   | Moved-by(X) | Moved-by(Y)
  ================================================================================
  MASTER_CLONE(I_MC_1)         |             |    Aligned |             |             |
  +I_MC_0                      |     MY      |    Aligned |             |             |
  MC(I_A/A_MC)                 |             |    Aligned |             |             |
  +I_B/B_MC                    |     R0      |    Aligned |     -0.2000 |      0.0000 |
  K(k4)                        |             |    Aligned |             |             |
  +k3                          |     MY      |    Aligned |      0.2000 |     -0.4450 |
  +k2                          |     R0      |    Aligned |      0.0000 |      0.0600 |
  +k1                          |     R90     | Infeasible |             |             |
  b(b1)                        |             |    Aligned |             |             |
  +b3                          |     R0      |    Aligned |      0.0000 |      0.1800 |
  +b2                          |     R0      |    Aligned |      0.0000 |     -0.3250 |
  c(c1)                        |             |    Overlap |             |             |
  S(s1)                        |             |    Overlap |             |             |
  ================================================================================
  ```

- The following command snaps clones of module eug to Master's power grid:
  ```
  align_partition_clones eug
  ```
- The following command aligns with all allowed tracks and automatic detection of pg layers, nets, and pitch of pg grid
  ```
  align_partition_clones eug -pg_horizontal_grid -pg_vertical_grid
  ```
- The following command aligns with all tracks and automatic detection of pg nets and pitch of pg grid
  ```
  align_partition_clones eug -pg_horizontal_grid -pg_layers {2 4 6} -pg_vertical_grid -pgLayer {3 5 7}
  ```
- The following command aligns with all tracks and automatic detection of pg pitch of pg grid
  ```
  align_partition_clones eug -pg_horizontal_grid -pg_layers {2 4 6} -pg_net VSS -pg_layers {4 6} -pg_net VDD -pg_vertical_grid -pg_layers
  {3 5} -pg_net VSS -pg_layers {5 7} -pg_net VDD
  ```
- The following command aligns with all allowed tracks and specified pg grid
  ```
  align_partition_clones eug -pg_horizontal_grid -pg_layers {2 4 6} -pg_net VSS -pg_pitch 1.48 -pg_net VDD -pg_pitch 1.28 -
  pg_vertical_grid -pg_layers {3 5 7} -pg_net VSS -pg_pitch 1.48 -pg_net VDD -pg_pitch 1.28
  ```

Related Information

- commit_partitions
- check_pin_assignment
- Partitioning the Design chapter in the *Innovus Stylus Common UI User Guide*

align_pins

```
align_pins
[-help]
[-ignore_pin_shapes]
[-mark_fixed]
[-move_fixed_pin]
[-keep_layer | -new_layer layer]
[-legalize]
[-no_snap]
[-pins pin_list]
[-reference ref_obj_name]
[-reference_type {partition_pin hinst}]
[-target ptn_name]
[-verbose]
```

Aligns pins between blocks on their facing edges. By default, this command snaps pins to the routing grid. You can use this command after partitioning floorplanning.

Notes:

- Specified pins will not be aligned if they do not have the reference connections on the reference block, or if the reference pins are not assigned prior to running this command.
- If this command is run on a blackbox that has one or more pins with a single LEF port but with multiple shapes within the LEF port, the command will remove these multiple shapes and create a single shape. If this command is run on a blackbox that has one or more pins with multiple LEF ports, the first LEF port will be processed as per the command and the remaining LEF ports will remain unchanged.
- This command supports Multi Level Hierarchical Flow.

The pin alignment priority is as follows:

- If you specify the `-reference` and the `-target` parameters but not the `-pins` parameter, the `align_pins` command aligns *all* pins of specified instances that are connected to the referenced object.
- If you specify the `-reference` parameter but not the `-target` and the `-pins` parameter, the `align_pins` command aligns *all* pins of all partitions that are connected to the referenced object.
- If you specify the `-reference`, `-target`, and the `-pins` parameters, the `align_pins` command tries to align *all* possible pins that can be aligned.
- If you do not specify the `-reference` parameter, the following order of priority will be used for aligning partition pins that have multiple connections:
  - Hard macro pins
  - I/O pads or pins
  - Partition pins
  - Standard cell pins

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `align_pins` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man align_pins`
`-ignore_pin_shapes`	Specifies that the aligned pin will ignore matching of shape with the reference pin. *Data_type*: bool, optional
`-keep_layer`	Specifies whether to keep the current pin layer. *Default*: If you do not specify this option, the new pin layer will be the same as the referenced layer to reduce wire routing. *Data_type*: bool, optional
`-legalize`	Specifies that the `legalize_pins` command should run automatically after the `align_pins` command.
`-mark_fixed`	Specifies that the pins that are moved by this command will have a *Fixed* status. *Default*: By default, the pins moved by this command have a *Placed* status. *Data_type*: bool, optional
`-move_fixed_pin`	Specifies that the pins with a *Fixed* status can also be moved. *Data_type*: bool, optional
`-new_layer` *layer*	Specifies a new pin layer. **Note:** Layers can be specified using the LEF layer names or layer ID numbers. *Data_type*: string, optional
`-no_snap`	Disables pins from being snapped to the routing grid. *Data_type*: bool, optional
`-pins` *pin_list*	Specifies the pin(s) to be aligned between the facing edges of the target block and reference block. You can use wildcards (`*?`) with this parameter. If you specify more than one pin, you must enclose them in curly braces. *Data_type*: string, optional
`-target` *ptn_name*	Specifies the target instance where its pins will be aligned to the pins of the specified instance. *Data_type*: string, optional
`-reference` *ref_object_name*	Specifies the reference object where the specified pins of the target block will be aligned to the pins of the reference object. The reference object can be a standard cell, an I/O pin, an I/O pad, a macro, or all I/O pins in the design by specifying the top cell as the reference object. Now the `align_pins` command can also take uncommitted ptn as the reference object. **Note:** You can also use wildcard (* or ?) for specifying the reference object names. *Data_type*: string, optional
`-reference_type {partition_pin hinst}`	Specifies the type of reference objects to consider. You can specify partition pins or hinsts. *Data_type*: enum, optional
`-verbose`	Generates intermediate messages. *Data_type*: bool, optional

Examples

- The following command aligns `A0` and `A1` pins of `p1` to the reference pins of `blockA`:

```
align_pins -reference blockA -target p1 -pins {A0 A1}
```

- Assume the following netlist:

```
module testcase (in1, in2, in3);
input in1, in2, in3;
wire net;
a i_a(.in1(in1), .in2(in2), .in3(in3), .out1(net));
```

```
BUFX1 i0(.A(net));
endmodule

module a(in1, in2, in3, out1);
input in1, in2, in3;
output out1;
endmodule
```

The following command aligns pins `in1`, `in2`, and `in3` on partition `i_a`:

```
align_pins -reference testcase -target i_a
```

```
  Moving pin [in1] of partition [a] connected to pin [in1] of top cell [testcase] to ALIGNED location ( 0.000 21.000 Metal3).
  Moving pin [in2] of partition [a] connected to pin [in2] of top cell [testcase] to ALIGNED location ( 0.000 38.920 Metal5).
  Moving pin [in3] of partition [a] connected to pin [in3] of top cell [testcase] to ALIGNED location ( 0.000 57.960 Metal3).
  Aligned 3 pin(s) of the partition [a] successfully.
```

Note: Only 3 of the 4 partition pins are moved because pin `out1` is connected to a nearby instance rather than a top-level IO pin.

Related Information

- `legalize_pins`
- `assign_partition_pins`
- `assign_io_pins`
- `commit_partitions`
- Partitioning the Design chapter in the *Innovus Stylus Common UI User Guide*
  - Validating Pin Placement Results
  - Aligning Partition Pins

assemble_design

```
assemble_design
[-help]
[-keep_pg_pin_geometry]
[-out_file fileName]
[-keep_block_halo]
[-skip_cells] [-skip_components] [-skip_nets] [-skip_special_nets]
{{[-top_dir dirname [-mmmc_file mmmcFile] [-cpf_file cpfFile] ]
  {[-block_dir dirname [[-encounter_format ]| [-floorplan_only ]]][-block_data {defFileName netListFileName} |
   {-floorplan_only | -encounter_format } ] [-no_def_merge ]}}|
   {[-top_design {topLib topCell topView} [-mmmc_file mmmcFile] [-cpf_file cpfFile] ] {[-oa_block {blockLib blockCell blockView} | -
block_cell blockCell ]
  [-all_timing_blocks [-ignore_timing_blocks {list_of_blocks}] [-report_only ]]}} }
[-encounter_format | -floorplan_only | -no_def_merge ]
[-keep_pin_geometry ]
```

Brings back specified block data to the top-level design for chip assembly. You should run this command from the directory with the full chip-level floorplan for the top-down hierarchical flow. You can run this command for designs that use the FE database as well as for designs that use the OpenAccess database. The `assemble_design` command supports mixed-origin design. That is, all partitions can have a design origin at the center, lower-left, any location inside the design boundary, or any location outside the design boundary. You can use this command after implementing blocks and top-level designs.

There are two main modes of operation - full and incremental.

- Full mode: When operating in full mode, a session is initiated with the assemble_design command which first loads the top-level design with blocks represented as hard macros then each block is merged within the full-chip context.
- Incremental mode: When operating in incremental mode, a top-level design is loaded prior to calling `assemble_design` which then merges the contents of one or more blocks.

Notes:

- The `assemble_design` command supports designs saved in the Innovus database format as well as OpenAccess designs.
- The `assemble_design` command supports the incremental capability which handles master/clones partitions at different levels of hierarchy and supports nested partitions. The incremental `assemble_design` capability enables partitions to be brought back in one session, thus, avoiding the need to run multiple sessions for bringing back all the data of a design. You can call the incremental `assemble_design` for a block multiple times in the same session without having to partition it.
- The `assemble_design` command supports mixed-origin design. That is, all partitions can have a design origin at the center, lower-left, any location inside the design boundary, or any location outside the design boundary. You can use this command after implementing blocks and top-level designs.
- The `assemble_design` command preserves Interface Logic Model (ILM) information by automatically bringing back ILMs from block level design to full-chip level design. For DEF data, ILMs can only be brought back if the `-block_dir` option is specified.
- The `assemble_design` command supports mixed data types. While specifying partition blocks, you can have different types of data for different partition blocks.
- The `assemble_design` command supports partitions with flexmodels. It re-assembles partition blocks with flexmodels back to the top-level design.
- The `assemble_design` command can read design information from the global init_* variables saved in the *.global file.
- The `assemble_design` command internally calls the `commit_global_net_rules` command that applies or restores the global net connectivity rules to the design and creates the necessary connections between instances and global nets. Consequently, the `assemble_design` command automatically generates a database where the global nets are already connected. This helps to ensure that all instance are correctly tied to PG nets.
- The `assemble_design` command internally checks color issues and uses the shift-mask functionality to assign correct colors to objects. As a result, at the end of `assemble_design`, the objects are already given the mask numbers (colorized) and the design has less color violations.
- The `assemble_design` command compares the properties of each block and top and uses a specific resolution for each property.
- The `assemble_design` command (DEF based) supports multi-threading. It honors the settings of the `set_multi_cpu_usage -local_cpu` parameter that is used to specify the number of CPUs on the local machine and is required for multi-threading.

  Note: This capability is not supported by the `-encounter_format` based `assemble_design`.

During `assemble_design` all rows are cut at the top and only the rows from inside the block are brought at the time of assembly. The `check_place` command does not report any false placement violations. Only in cases where partitions are rotated R90 (like vertical rows) are the cells marked as fixed.

A summary report is generated at the end of each `assemble_design` run which helps in debugging issues. The generated report includes information like the run-time for `assemble_design`, number of partitions and their instance count.This report is useful in identifying potential issues in the flow.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `assemble_design` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man assemble_design`
`-all_timing_blocks`	Enables automatic detection and assembly of blocks with digital content. The `-block_cell` option can be used to manually force blocks to be assembled, and `-except_blocks` option can be used to exclude cells from being assembled as it overrides to the default list of blocks to be assembled. *Data_type*: bool, optional
`-block_cell` *blockCell*	Specifies the cell name for the block cell. *Data_type*: string, optional
`-block_data` *defNamenetlistName*	
	Specifies the DEF file name and the Verilog netlist file name. Use this parameter if the Verilog netlist file name or the DEF file name is different from the cell or module name. You must provide the full path (absolute or relative) with the file names. **Note:** If you specify more than one block-level design, specify the `-block_data` option separately for each block-level DEF file and Verilog netlist file. For example: `-block_data defFile_1 VerilogFile_1 -block_data defFile_2 VerilogFile_2` *Data_type*: string, optional
`-block_dir` *dirname*	Specifies the path to the block-level design(s). **Note**: If you specify more than one block-level design, specify the -block_dir option separately for each block-level directory. For example: `-block_dir ptnDir1 -block_dir ptnDir2` *Data_type*: string, optional
`-cpf_file` *cpfFile*	Specifies the name of the (Chip Level) CPF file that contains the power intent and timing information. *Data_type*: string, optional
`-encounter_format`	Specifies that the Innovus place-and-route data should be used for `-block_dir` option. **Note**: The `-encounter_format` parameter cannot be used with the `-block_data` parameter because the `-encounter_format` parameter points to data from the Innovus database format whereas the `-block_data` parameter points to data from Verilog netlist and DEF name. *Default*: By default, data from DEF files is used. *Data_type*: bool, optional
`-floorplan_only`	Specifies that only floorplan data should be assembled. When specified it brings back the floorplan changes from a partition block to full-chip level design. It supports un-committed partition(s) and replaces the top level uncommitted partition(s) with updated block floorplan data. **Note:** After `assemble_design -floorplan_only` command, any signal net at top-level design may be overlapped with other floorplan objects. You may need to reroute or ECO route the design. Additionally, PG net of CHIP, place, and route data may not be brought back.
`-ignore_timing_blocks {`*list_of_blocks*`}`	
	Specifies the list of block names to be excluded from being assembled even if they contain digital content. This option only applies to the case where `-all_timing_blocks` is used. If not specified, there is no restriction on the detection and assembly that is automatically determined. *Default*: `""` *Data_type*: string, optional
`-keep_block_halo`	When specified, the `assemble_design` command preserves the block place and route halo. *Data_type*: bool, optional
`-keep_pg_pin_geometry`	Retains the partition PG pin geometry. *Data_type*: bool, optional

`-keep_pin_geometry`	Specifies that for every partition pin, a DRCFILL metal of the type special route is placed on top of the partition pins. *Data_type*: bool, optional
`-mmmc_file mmmcFile`	Specifies the name of the (Chip Level) MMMC file that is used to supply the MMMC information. Normally, it is a chip level file named *viewDefination.tcl*. *Data_type:* string, optional
`-no_def_merge`	Specifies that the DEF merge capability will be disabled and normal `assemble_design` will be used. *Data_type*: bool, optional
`-oa_block blockLib blockCell blockView`	
	Specifies the block-level library, cell, and view names respectively for a design that was saved into an OpenAccess database. This parameter is used when bringing back block data for the partition from an OpenAccess database. **Note:** If you specify more than one block-level design, specify the `-oa_block` option separately for each block. For example: `-oa_block libForOA TDSP_CORE ptnView1 -block libForOA TDSP_ARB ptnView1` *Data_type*: string, optional
`-out_file fileName`	Specifies name of the output file. *Default*: `./<design>.assembleDesign.rpt` *Data_type*: string, optional
`-report_only`	When enabled, the `assemble_design` command only reports the list of blocks that would be assembled and does not perform actual assembling. **Note:** The `-report_only` parameter can only be used in the `assemble_design` command along with the `-all_timing_blocks` option, to get a list of (and not actually assemble) blocks that would be assembled by a subsequent `assemble_design -all_timing_blocks` command invocation without the `-report_only` option. This list of blocks can be used to include or exclude blocks to be assembled via the `-block_cell` or `-ignore_timing_blocks` options. *Data_type*: bool, optional
`-skip_cells`	Skips merging standard cells. It does not load the standard cells section from DEF. *Data_type*: bool, optional
`-skip_components`	Skips merging all components.It does not load the COMPONENT section from DEF. *Data_type*: bool, optional
`-skip_nets`	Skips merging signal routes. It does not load the NETS section from DEF. *Data_type*: bool, optional
`-skip_special_nets`	Skips merging special routes. It does not load the SPECIALNETS section from DEF. *Data_type*: bool, optional
`-top_design topLib topCell topView`	
	Specifies the top-level library, cell, and view names respectively for a design that was saved into an OpenAccess database. This parameter is used when bringing back top-level data from an OpenAccess database. *Data_type*: string, optional
`-top_dir dirname`	Specifies the path to the top-level design. This design directory should contain the Verilog netlist and DEF files. *Data_type*: string, optional

Examples

- The following command assembles the design after bringing back information from the top-level cell and two block-level cells. The top-level design data is in the `dirTop` directory and the block-level data is in the `dirBlock1` and `dirBlock2` directories.

  ```
  assemble_design -top_dir dirTop -block_dir dirBlock1 -block_dir dirBlock2
  ```

- The following command assembles a design with the following block-level information:
  - the DEF file with the name `def_file`

    - the verilog netlist v_netlist.v

      ```
      assemble_design -top_dir dirTop -block_data def_file v_netlist.v
      ```

- The following command assembles a design that was saved in an OpenAccess database. The design contains two blocks.
    - The library, cell, and view names for the top-level are `libForOA`, `DTMF`, and `ptnView1` respectively.
    - The library, cell, and view names for the first block are `libForOA`, `TDSP_CORE`, and `ptnView1` respectively.
    - The library, cell, and view names for the second block are libForOA, TDSP_ARB, and ptnView1 respectively.

      ```
      assemble_design -top_design {libForOA DTMF ptnView1} -oa_block {libForOA TDSP_CORE ptnView1} -oa_block {libForOA TDSP_ARB ptnView1}
      ```

- The following command is an example of incremental assemble design capability that can restore master and clones at different levels of hierarchy. Along with assembling nested partitions:
    - Block1 and Block2 are partitions at top level
    - Block1 contains Block3 and Block4
    - Block4 is a clone of Block2

      ```
      read_db topDesign.enc.dat
      assemble_design -block_dir Block1.enc.dat -block_dir Block2.enc.dat -block_dir Block3.enc.dat
      ```

- `-mmmc_file` and `-cpfFile` do not work with incremental `assemble_design`, however in the following command they will work:

  ```
  assemble_design -top_dir dirTop -block_dir dirBlock1 -mmmcFile  viewDefination.tcl. -cpf_file cpfFile
  ```

Related Information

- `commit_global_net_rules`
- `check_place`
- `commit_partitions`
- `write_db`
- Partitioning the Design chapter in the *Innovus Stylus Common UI User Guide*
    - Flow Methodologies
    - Handling of Blackboxes with Non-R0 Orientation
    - Working with OpenAccess Database

assign_io_pins

```
assign_io_pins
[-help]
[-align]
[-debug_pin_all]
[-debug_pin_file fileName]
[-improve_si]
[{-pins { pin_name_list } | -pin_file  filename  | -exclude_pin_file  filename } [-move_fixed_pin ] [ -ignore_group_pins ]]
[-auto_bus_group ]
```

Assigns the top I/O pin locations for a block-level design. Set the maximum or minimum pin layers with `assign_pins_max_layer or assign_pins_min_layer` attributes. By default, the `assign_io_pins` command sets the depth of all partition pins equal to: MINAREA/pin width. This ensures that the minimum area rule is met. However, if any pin-size-related constraints are specified on these pins, the user-specified pin size will have a higher precedence.

Notes:

- The MINAREA/pin width means that the default pin width will be met MAR (minimum area rule) if an area rule of this layer is defined in the technology LEF. An I/O pin width value is based on the following order:
  1. User-specified pin width constraint.
  2. Minimum area rule of the layer if it is available in the technology file.
  3. Default layer width
- When assigning pins, the `assign_io_pins` command does not check for violations between placed pins and shapes of objects (such as macros, standard cells, etc) just inside the partition fence (design boundary). This may result in DRV violations if the fence (design boundary) to core distance is zero.
- The `assign_io_pins` command honors preferred routing layer attributes as soft constraints during pin assignment and reports the nets that have these attributes. It also honors the *keep out spacing* constraints on pin groups and net groups.
- In bottom-up flow, the `assign_io_pins` command promotes both signal and P/G pins from the list of pins promoted by the `set_promoted_macro_pin` command.
- **Note**: The `assign_io_pins` command honors net/pin groups that have all nets from multiple buffer chains added in the same group. Such groups can be generated through bus guide flow, feedthrough insertion flow, or can be user created. It places the pins of such nets in order to achieve better pin alignment and improve the QoR.

You can use this command after running placement and early global routing to legalize the I/O pins. However, placement is not a mandatory step for pin assignment. Placement is recommended for better QOR.

Parameters

`-align`	If you specify this option, each top I/O pin will be aligned with one of the instance pins to which it is connected. The instance pin with which the I/O pin is aligned is the one whose distance from its nearest edge is shortest among all the instance pins connected to the top I/O pin. *Default*: If you do not specify this option, the location of the top pin is based on the average of the distance of all instance pins (that are connected to the top I/O pin) from their respective nearest edges. *Data_type*: bool, optional
`-auto_bus_group`	Specifies that the buses will be grouped automatically. **Note:** Internally the `create_pin_group` command is applied on buses. It is removed automatically after the command execution. *Data_type*: bool, optional

`-debug_pin_all`	Specifies that all pins will be debugged. **Note:** The `-debug_pin_all` and `-debug_pin_file` parameters can be used to write the following information in the log file: • Pin assign mode information • Connectivity information of each partition pin, including pinGroup/netGroup name and NDR name • Available pin slot information on each edge for each layer and pin slot density • PinGuide box and virtual pinGuide box information, master-clone series, feedthru chain and so on. • Reason for failure on the first failed location • For pins with location constraints, if the pin is not placed at the location constraint then the reason is provided in a tabular form. • For placed partition pins, the pin location, layer, edge, spacing and so on are provided. *Data_type*: bool, optional
`-debug_pin_file` *filename*	Specifies the file containing the pin-list for partition or the net that is to be used for debugging. *Data_type*: string, optional
`-exclude_pin_file` *filename*	Specifies pin-list to be excluded for partition in a file. *Data_type*: string, optional
`-help`	Outputs a brief description that includes the type and default information for each `assign_io_pins` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man assign_io_pins`
`-ignore_group_pins`	Avoid placing the whole group of pins. With this parameter enabled, the `pin_group` and `net_group` pins are ignored during selected pin assignment. *Data_type*: bool, optional
`-improve_si`	Avoid signal integrity issues by blocking tracks while placing pins to avoid overlap of wires. With this parameter specified, the `assign_io_pins` command does not allow pins to be placed on the tracks in an area which spans a layer higher or lower (on the preferred routing direction) and also on the tracks on the left and right of the placed pin. *Data_type*: bool, optional
`-move_fixed_pin`	If you use this parameter, the specified pins with a *`Fixed`* status can also be moved. However, this parameter must be used with the -pins parameter to specify the list of fixed pins to move while assigning io-pins. **Note:** Pins that are not specified but have a *`Fixed`* status are not moved. *`Default`* : If you do not specify this parameter, the specified pins with a *`Fixed`* status are *not* moved. *Data_type*: bool, optional
`-pins` *pin_name_list*	Specifies that only the specified pins should be assigned. **Note:** To move fixed pins while assigning io pins, you must specify the `-move_fixed_pin` parameter. *`Default`* : If you do not specify this parameter, all pins are assigned. *Data_type*: string, optional

`-pin_file` *`filename`*	Specifies the pin list for a partition in a file.
	In the pin file, the pins and nets are specified in the following format:
	`Partition:` *`PtnName/BlockName`*
	`Pin1`
	`Pin2`
	`..`
	`PinN`
	`Partition:` *`PtnName/BlockName`*
	`Pin1`
	`Pin2`
	`..`
	`PinN`
	`Nets:`
	`netA`
	`netB`
	`..`
	`netN`
	Note: While specifying the pin list in the pin_file, it is necessary to give the “Partition” keyword followed by the name of the partition or block. If this is not specified, no operations are performed. You can specify multiple partitions in the same pin_file.
	The pin_file does not support buses or wild cards for pin names. If a pin specified in the list or a partition is not present in the design, a warning is displayed.
	Data_type: string, optional

Example

- The following command assigns all io pins and can also move pins with fixed-status.

  ```
  assign_io_pins -move_fixed_pin -pins *
  ```

- The following command accept a file *`pinLst.txt`* that contains the list of pins and assigns io pins.

  ```
  assign_partition_pins -pin_file pinLst.txt
  ```

  Where *`pinLst.txt`* contains:

  ```
  Partition: top
  out1
  out2
  out3
  Partition: A
  in1
  in2
  in3
  Partition: B
  pina
  pinb
  pinc
  ```

  The `assign_io_pins` command selects all Io pins under the section `Partition:top`.

Related Information

- `create_pin_group`
- `update_pin_group`
- `create_pin_guide`
- `create_net_group`
- `assign_partition_pins`
- `report_pin_assignment_statistics`
- `compare_pin_assignment_statistics`
- `floorplan` Category Attributes
  - specifically `floorplan_snap_io_grid`
- `assign_pins` Category Attributes
- Partitioning the Design chapter in the *Innovus Stylus Common UI User Guide*
  - Block Implementation
  - Assigning I/O Pins

assign_partition_pins

```
  assign_partition_pins
[-help]
[-based_on_master_only]
[-blackbox]
[-debug_pin_all]
[-debug_pin_file filename]
[-write_virtual_pin_guide]
[-ignore_abutted_check]
[-improve_pin_order]
[-improve_si]
[-layer_priority {layeridList}]
[-mark_fixed] [-pin_movement_statistics]
[-skip_auto_bus_bit_order]
[-skip_place_external_unconnected_pin]
[-unplaced_only]
[ {{{-partitions partition_name  {-pins {pin_name list}}} |
 {-pin_file filename [-exclude_partitions ptn_name]} |
 -exclude_pin_file filename}
[-move_fixed_pins ] [ -ignore_group_pins ]}]
[-align_feedthru_pins_opt_with_master_clone ]
[-enforce_route | -enforce_route_legal | -enforce_flightline ]
[-auto_bus_group ]
```

Assigns partition pins before the partitions are committed. By default, the assigned pins will be in a Placed status after running this command. You can specify a Fixed status by running this command with the `-mark_fixed` option. The `assign_partition_pins` command does not align pins in the local context thus avoiding mis-aligning in other scenarios. minimizing maximum mis-alignment effects across farsighted net pairs (2-pins nets). For example, in master/clone pin assignment, since the clone pin follows the master's position, the position of the pin on master partition is determined such that pin on clone also gets aligned on clone.

Running the `write_floorplan` command saves the pin assignment in the floorplan to a separate file named `fpFileName.fp.ptnpin`, which is included in main `fpFileName.fp` floorplan file.

By default, the `assign_partition_pins` command sets the depth of all partition pins equal to: MINAREA/pin width. This ensures that the minimum area rule is met. However, if any pin-size-related constraints are specified on these pins, the user-specified pin size will have a higher precedence. The `assign_partition_pins` command internally calls the `report_unaligned_nets` command to generate the summary report at the command line. When assigning pins, the `assign_partition_pins` command does not check for violations between placed pins and shapes of objects (such as macros, standard cells, etc) just inside the partition fence (design boundary). This may result in DRV violations if the fence (design boundary) to core distance is zero. You can use this command after specifying the partitions, placing the design, and after running or loaded routing information. If no routing information is available, the Innovus software uses the projected pin assignment. However, placement is not a mandatory step for pin assignment. Placement is recommended for better QOR.

Notes:

- The `assign_partition_pins` command honors preferred routing layer attributes as soft constraint during pin assignment and reports the nets that have some attributes assigned to them. It also honors the keep out spacing constraint for a pin group or net group.
- This command only assigns signal pins. Power and ground pins are created during partition. However, this pin optimization honors power stripes and follow pins.
- This command assigns pins for blackboxes as well as regular partitions. If this command is run on a blackbox that has one or more pins with a single LEF port but with multiple shapes within the LEF port, the command will remove these multiple shapes and create a single shape. If this command is run on a blackbox that has one or more pins with multiple LEF ports, the first LEF port will be processed as per the command and the remaining LEF ports will remain unchanged.
- The `assign_partition_pins` command performs automatic pin assignment of partitions inside other partitions. It places pins of nested partitions similar to single level partitions.
- The `assign_partition_pins` command honors net/pin groups that have all nets from multiple buffer chains added in the same group. Such groups can be generated through bus guide flow, feedthrough insertion flow, or can be user created. It places the pins of such nets in order to achieve better pin alignment and improve the QoR.
- The `assign_partition_pins` command can place the pins on the partitions that are connected to PG supplies defined in the netlist.

Parameters

`-align_feedthru_pins_opt_with_master_clone`	Specifies that the pins should be placed such that they are aligned on different partitions across buffers on master/clone partitions.
	• In symmetric designs, since there is no conflict with master/clone so it aligns the pins of all the path.
	• In non-symmetric designs, it will try to place pins such that master-clone alignment count should not degrade.
	  ◦ It only assigns a feedthru path if there is common alignment zone available between master-clone alignment and feedthru chain alignment.
	  ◦ It assigns pins in common alignment zone for first pins in feedthrough path.
	Data_type: bool, optional

`-auto_bus_group`	Specifies that the buses will be grouped automatically. **Note:** Internally the `create_pin_group` command is applied on buses. It is removed automatically after the command execution. *Data_type*: bool, optional
`-based_on_master_only`	Specifies that for designs that have partition clones, the command should consider only the partition master and use the master assignment for all the clones. *Default*: For designs that have partition clones, this command by default also considers partition clones while assigning pins. *Data_type*: bool, optional
`-blackbox`	Assigns pins of blackboxes. *Data_type*: bool, optional
`-debug_pin_all`	Specifies that all pins will be debugged. **Note:** The `-debug_pin_all` and `-debug_pin_file` parameters can be used to write the following information in the log file: • Pin assign mode information • Connectivity information of each partition pin, including pinGroup/netGroup name and NDR name • Available pin slot information on each edge for each layer and pin slot density • PinGuide box and virtual pinGuide box information, master-clone series, feedthru chain and so on. • Reason for failure on the first failed location • For pins with location constraints, if the pin is not placed at the location constraint then the reason is provided in a tabular form. • For placed partition pins, the pin location, layer, edge, spacing and so on are provided. *Data_type*: bool, optional
`-debug_pin_file` *filename*	Specifies the file containing the pin-list for partition or the net that is to be used for debugging. *Data_type*: string, optional
`-enforce_flightline`	Specifies that the pin assignment completely follows fly lines, without taking into consideration any pin constraints. *Data_type*: bool, optional
`-enforce_route`	Specifies that the pin assignment completely follows the routing results generated by Early Global Route, without taking into consideration any pin constraints. For example, the following constraints are not considered when the `-enforce_route` parameter is used: • Constraints related to pin spacing, layer, width, depth, and side • Pin Guide constraints and Pin Group constraints The `-enforce_route` parameter can be useful when, for example, you want to compare pin locations based purely on Early Global Route results with pin locations that honor the specified pin constraints. *Data_type*: bool, optional

`-enforce_route_legal`	Specifies that during pin assignment, the pins are placed close to the routing cross points but at legal locations only. consequently all the pins are legally placed. The `-enforce_route_legal` parameter ignores the pin_group, net_group, and pin_guide constraints. *Data_type*: bool, optional
`-exclude_partitions` *ptn_name*	Specifies pin-list to be excluded for a partition in a file. In the pin file, the pins and nets are specified in the following format: `Partition:` *PtnName/BlockName* *Pin1* *Pin2* .. *PinN* `Partition:` *PtnName/BlockName* *Pin1* *Pin2* .. *PinN* `Nets:` *netA* *netB* .. *netN* *Data_type*: string, optional
`-exclude_pin_file` *filename*	Specifies pin-list to be excluded for partition in a file. *Data_type*: string, optional
`-help`	Outputs a brief description that includes type and default information for each `assign_partition_pins` parameter. For a detailed description of the command and all of its parameters, use the man command: `man assign_partition_pins`.
`-ignore_abutted_check`	When enabled, the abutted check is ignored during pin assignment. This results in faster pin assignment on abutted designs. *Data_type*: bool, optional
`-ignore_group_pins`	Avoid placing the whole group of pins. With this parameter enabled, the `pin_group` and `net_group` pins are ignored during selected pin assignment. **Note:** If all the pins in a group have been individually selected for pin assignment and `-ignore_group_pins` is applied then the pin assignment takes precedence and `-ignore_group_pins` parameter is not honored for this group. Rest of the groups whose pins appear in the selection list continue to be ignored. *Data_type*: bool, optional
`-improve_pin_order`	Reduces flightline crossing during pin alignment. The partition and the blackbox pins are rearranged to be aligned with external connections: • Guided pins are reordered within the pin guide box. • Pins are rearranged at (or close to) their original locations. • Pin ordering is performed independently for each partition edge. The locations of fixed pins, aligned pins or multi-location pins is not changed. Master-clone pins are not reordered. **Note:** `-improve_pin_order` is a post processing step during pin assignment so it can override the pin assignment that is complying with `-max_pin_movement_for_align`. This option should not be used with `-max_pin_movement_for_align`, if you strictly want to control the maximum pin movement for alignment. *Data_type*: bool, optional
`-improve_si`	Avoid signal integrity issues by blocking tracks while placing pins to avoid overlap of wires. With this parameter specified, the `assign_partition_pins` command does not allow pins to be placed on the tracks in an area which spans a layer higher or lower (on the preferred routing direction) and also on the tracks on the left and right of the placed pin. *Data_type*: bool, optional

`-layer_priority {layeridList}`	Specifies the metal layer(s) that should be used with priority during pin assignment such that there is an increase in alignment in the prioritized layers. With this parameter specified, there is no degradation in the total unaligned nets of the partition/design. There is an increase in the unaligned number in the prioritized layers and a decrease in the over all pin count in the non-prioritized layers. **Note**: The list of layers to be given priority must be a sunset of the layers selected with `assign_pins_max_layer` and `assign_pins_min_layer` attributes. **Note**: In case of conflict in the layer priority between the `assign_partition_pins` command and the `create_pin_guide` command on some edges, the `create_pin_guide` command priority takes precedence. For example, in the following command: `assign_partition_pins -layer_priority {7 6 2 3}` • Most aligned pins are in layers 7 and 3 or 6 and 2 (horizontal / vertical) • Most unaligned pins are also in 7,6,2,3 • Maximum pins are in these layers *Data_type*: string, optional
`-mark_fixed`	Specifies that all pins will have a *Fixed* status. *Data_type*: bool, optional
`-move_fixed_pins`	If you use this parameter, the specified pins with a *Fixed* status can also be moved. **Note:** Pins that are not specified but have a *Fixed* status are not moved. *Default*: The specified pins with a *Fixed* status are not moved. *Data_type*: bool, optional
`-partitions partition_name`	Specifies the partition names for pin assignment. *Default*: Pins are assigned for all partitions. **Note:** Pin assignment for a selected partition by specifying the `partitionName` is obsolete and will be removed in the next major release of the software. The obsolete parameter still works in this release, but to avoid warnings, and to ensure compatibility with future releases, update your scripts to use `-partitions partition_name -pins pin_list ...` *Data_type*: string, optional

`-pin_file filename`	Specifies the pin list for a partition in a file.
	In the pin file, the pins and nets are specified in the following format:
	`Partition: PtnName/BlockName`
	`Pin1`
	`Pin2`
	`..`
	`PinN`
	`Partition: PtnName/BlockName`
	`Pin1`
	`Pin2`
	`..`
	`PinN`
	`Nets:`
	`netA`
	`netB`
	`..`
	`netN`
	Note: While specifying the pin list in the pin_file, it is necessary to give the “Partition” keyword followed by the name of the partition or block. If this is not specified, no operations are performed. You can specify multiple partitions in the same pin_file.
	The pin_file does not support buses or wild cards for pin names. If a pin specified in the list or a partition is not present in the design, a warning is displayed.
	Data_type: string, optional

`-pins {pin_name list}`	Specifies the list of pins to be assigned in the respective partitions. *Default*: Pins are assigned for all partitions. *Data_type*: string, optional

`-pin_movement_statistics`	Prints a summary of the deviation (in micrometers) of the final pin locations from the point of intersection of the net and the partition edge.
	The report on the screen contains the following statistics:
	• Maximum displacement on the X axis. This is the difference in the x-coordinates of the point of intersection of the net and the partition edge, and the point at which the pin is finally assigned.
	• Maximum displacement on the Y axis.
	• Maximum X+Y displacement. The X+Y displacement of a pin is the sum of the displacement of a pin on the X axis and the displacement of the pin in the Y axis. For example, if a pin has moved 10 micrometers on the X axis and 5 micrometers on the Y axis, the X+Y displacement for that pin is 15. Thus, if there are three pins and their X+Y displacements are 5, 23, and 30 respectively, the maximum X+Y displacement is reported as 30.
	• Mean of the X+Y displacement.
	The report appended to the Innovus log file contains the following additional details:
	• Names of the pins that have moved the maximum distances.
	• The coordinates of the point of intersection of the net and the partition edge, and the point at which the pin is finally assigned.
	Here is a sample report displayed on screen.
	`Partition: [tdsp_core]`
	`maximum X = 120.1200 um (Pin: t_data_out[6])`
	`maximum Y = 58.8000 um (Pin: rom_data_in[4])`
	`maximum X+Y = 154.5600 um (Pin: t_data_out[6])`
	`moved from (706.2000, 1025.0800) -> (586.0800, 990.6400)`
	`mean X+Y = 12.5600 um`
	`total pins = 114`
	`pins moved = 75`
	The log file has additional information such as:
	`Pin: int: X Movement: 3.3000 um ; Y Movement: 0.0000. From (701.2500, 1247.6800) -> (697.9500, 1247.6800)`
	Data_type: bool, optional

`-skip_auto_bus_bit_order`	Skips automatically placing the bus pins together. *Data_type*: bool, optional
`-skip_place_external_unconnected_pin`	Skips placing partition pins that are not connected outside the partition. *Data_type*: bool, optional
`-unplaced_only`	Assigns only unplaced pins. *Data_type*: bool, optional
`-write_virtual_pin_guide`	Writes the virtual pin_guide for net_group and pin_group. *Data_type*: bool, optional

Example

- The following command assigns partition pins of all partitions
  `assign_partition_pins`
- The following command assign partition pins, including those which have fixed status of all partitions and mark them as fixed after assigning them
  `assign_partition_pins -mark_fixed -move_fixed_pins`
- The following command accept a file *`pinLst.txt`* that contains the list of pins to be placed.

```
assign_partition_pins -pin_file pinLst.txt
```

Where *pinLst.txt* contains:

```
Partition: top
out1
out2
out3
Partition: A
in1
in2
in3
Partition: B
pina
pinb
pinc
```

The `assign_partition_pins` command, considers all pins of all partitions in the file, except the pins under the section Partition:top.

- The following command ignores the `partition B` mentioned in the pin file pin.list.
  ```
  assign_partition_pins -pin_file pinLst.txt -exclude_ptn {B}
  ```

Related Information

- `report_unaligned_nets`
- `assign_pins` Category Attributes
- `create_pin_guide`
- `create_pin_group`
- `read_partition_pins`
- `reset_partition_pins`
- `write_partition_pins`
- `report_pin_assignment_statistics`
- `compare_pin_assignment_statistics`
- Partitioning the Design chapter in the *Innovus Stylus Common UI User Guide*
  - Chip Planning
  - Blackbox Flow
  - Handling of Blackboxes with Non-R0 Orientation
  - Route-based Pin Assignment
  - Tips for Assigning Partition Pins
  - ECO Pin Assignment

assign_pins Category Attributes

The root attributes in the assign_pins category are:

- assign_pins_advanced_node_rule_support
- assign_pins_allow_non_ndr_net_pins_on_ndr_tracks
- assign_pins_allow_unconnected_in_abutted_edge
- assign_pins_blocked_boundary_macro_distance
- assign_pins_edit_in_batch
- assign_pins_force_abutment_with_fixed
- assign_pins_max_channel_width_as_abutted
- assign_pins_max_distance_pair
- assign_pins_off_stripe
- assign_pins_pin_to_stripe_distance
- assign_pins_pin_to_via_distance_non_preferred_direction
- assign_pins_pin_to_via_distance_preferred_direction
- assign_pins_promoted_macro_bottom_layer
- assign_pins_promoted_macro_top_layer
- assign_pins_restricted_boundary_macro_distance
- assign_pins_strict_abutment

To set the value of any root attribute, use the `set_db` command:

```
set_db attribute_name value
```

To get the current value of any root attribute, use the `get_db` command:

```
get_db attribute_name
```

To get the current value of all attributes in the `assign_pins` category, use the following command:

```
get_db -category assign_pins
```

assign_pins_advanced_node_rule_support

```
assign_pins_advanced_node_rule_support {true | false}
```

Default: `true`

Data_type: bool, read/write

A `true` value specifies that 20nm (and below) as well as DPT LEF rules will be respected (if present in LEF technology file) for determining and resolving any DRC violations, during all pin assignment related commands.

Related Commands

- `assign_io_pins`
- `assign_partition_pins`
- `align_pins`
- `edit_pin`

assign_pins_allow_non_ndr_net_pins_on_ndr_tracks

```
assign_pins_allow_non_ndr_net_pins_on_ndr_tracks {true | false}
```

Default: `true`

Data_type: bool, read/write

A `true` value allows general pins to be placed on NDR tracks (if available). Alternatively, if the value is set to false, the NDR-track rule is followed strictly and the net pins are placed on matching NDR tracks only.

Related Commands

- `assign_io_pins`
- `assign_partition_pins`
- `align_pins`
- `edit_pin`

assign_pins_allow_unconnected_in_abutted_edge

`assign_pins_allow_unconnected_in_abutted_edge {true | false}`

Default: `true`

Data_type: bool, read/write

A `true` value specifies that unconnected partition pins can be placed in abutted edges.

Related Commands

- `assign_io_pins`
- `assign_partition_pins`
- `align_pins`
- `edit_pin`

assign_pins_blocked_boundary_macro_distance

`assign_pins_blocked_boundary_macro_distance` *`distance`*

Default: 3

Minimum: 0

Maximum: 100

Data_type: float, read/write

Blocks automatic pin assignment from putting pins, on the part of the partition boundary facing the block, where the channel width between the partition boundary and the block is less than the value specified.
For channel width below the specified value, it blocks pins on the boundary facing this channel.

Related Commands

- `assign_io_pins`
- `assign_partition_pins`
- `align_pins`
- `edit_pin`

assign_pins_edit_in_batch

`assign_pins_edit_in_batch {true | false}`

Default: `false`

Data_type: bool, read/write

Enables pin-editing in batch mode.

Note: Once the interactive pin assignment is complete, it is important to set the mode to false. This can be done using the `set_db assign_pins_edit_in_batch false` command.

Note: The `assign_pins_edit_in_batch` attribute does not support the internal pins.

Related Commands

- `assign_io_pins`
- `assign_partition_pins`
- `align_pins`
- `edit_pin`

assign_pins_force_abutment_with_fixed

`assign_pins_force_abutment_with_fixed {true | false}`

Default: `true`

Data_type: bool, read/write

Specifies that for the connected pin of fixed abutted pins, ignore soft constraints check.

Some partitions in abutted design have preassigned pins with 1-track spacing and marked as fixed. As the design is abutted, there is no other legal location for such pins. Also, the automatic pin assignment has default spacing of two tracks. As a result, all pins connecting to these reference pins with one spacing are not assigned and lot of pins are left unassigned. When enabled, track spacing violations are ignored. As a result, assigns the corresponding pins opposite to the prefixed pins while maintaining abutment of pins.

Related Commands

- `assign_io_pins`
- `assign_partition_pins`
- `align_pins`
- `edit_pin`

assign_pins_max_channel_width_as_abutted

`assign_pins_max_channel_width_as_abutted` *`channel_width_in_microns`*

Default: 3

Minimum: 0

Maximum: 400

Data_type: double, read/write

Specifies that all channels with width less than or equal to the specified width will be considered as abutted and abutment rules will be applied on them.

Notes:

- The `assign_partition_pins`, assign_io_pins, and the `legalize_pins` commands do not place non-neighbor pins in the thin channels considered abutted.
- The check_pin_assignment command reports an abutment violation in case a non-neighboring pin is placed in the thin channel (abutted). The `-pre_check` option of the `check_pin_assignment` command that checks for the feasibility of pin assignment, also treats these thin channels as abutted.

Related Commands

- `assign_io_pins`
- `assign_partition_pins`
- `align_pins`
- `check_pin_assignment`
- `edit_pin`

assign_pins_max_distance_pair

`assign_pins_max_distance_pair` *`distance`*

Default: 50

Minimum: 0

Maximum: 10000

Data_type: double, read/write

Ignores moving corresponding connected pins of 2-pin-connection nets where distance (in microns) between the pins is equal or greater than distance specified. You can avoid moving the corresponding pin for any channel width (routing bends are acceptable in channels) by using the distance value for the `-max_distance_paring` parameter as 0.

Related Commands

- `assign_io_pins`
- `assign_partition_pins`
- `align_pins`
- `edit_pin`

assign_pins_off_stripe

`assign_pins_off_stripe {below | all | none}`

Default: `none`

Data_type: enums {below all none}, read/write

Prevents the generation of pins on metal layers with respect to the power and ground stripe.

- `below`: Prevents the generation of pins on metal layers below the power and ground stripes.
- `all`: Prevents the generation of pins on all metal layers, above and below the power and ground stripes.
- `none`: Allows the generation of overlapping pins on all metal layers.

Related Commands

- `assign_io_pins`
- `assign_partition_pins`
- `align_pins`
- `check_pin_assignment`
- `edit_pin`

assign_pins_pin_to_stripe_distance

`assign_pins_pin_to_stripe_distance {`*`listOfLayerDistanceValuePair`*`}`

Default: `{}`

Data_type: string, read/write

Specifies the distance (in microns) from pin to stripe. It is layer specific.

Related Commands

- assign_io_pins
- assign_partition_pins
- align_pins
- check_pin_assignment
- edit_pin

assign_pins_pin_to_via_distance_non_preferred_direction

`assign_pins_pin_to_via_distance_non_preferred_direction {listOfLayerDistanceValuePair}`

Default: {}

Data_type: string, read/write

Specifies the distance (in microns) from pin to via-stack. It is layer specific.

Related Commands

- assign_io_pins
- assign_partition_pins
- align_pins
- check_pin_assignment
- edit_pin

assign_pins_pin_to_via_distance_preferred_direction

`assign_pins_pin_to_via_distance_preferred_direction {listOfLayerDistanceValuePair}`

Default: {}

Data_type: string, read/write

Specifies the distance (in microns) from pin to via-stack from the partition fence boundary. It is layer specific.

Related Commands

- assign_io_pins
- assign_partition_pins
- align_pins
- check_pin_assignment
- edit_pin

assign_pins_promoted_macro_bottom_layer

`assign_pins_promoted_macro_bottom_layer {layername}`

Default: 31

Data_type: string, read/write

Specifies the minimum metal layer name for promoting macro pins.

Related Commands

- assign_io_pins
- assign_partition_pins

assign_pins_promoted_macro_top_layer

`assign_pins_promoted_macro_top_layer {layername}`

Default: `31`

Data_type: string, read/write

Specifies the maximum metal layer name for promoting macro pins.

Related Commands

- assign_io_pins
- assign_partition_pins

assign_pins_restricted_boundary_macro_distance

`assign_pins_restricted_boundary_macro_distance distance`

Default: `30.0`

Minimum: 0

Maximum: 100

Data_type: double, read/write

Blocks automatic pin assignment, from putting pins on the part of the partition boundary facing the block, in the places where the channel width between the partition boundary and the block is less than the specified value.
For the channel width below the specified value, allowed layers and the channel length/width will be considered to allow a suitable number of pins on the boundary facing this channel.

Related Commands

- assign_io_pins
- assign_partition_pins
- align_pins
- check_pin_assignment
- edit_pin

assign_pins_strict_abutment

`assign_pins_strict_abutment {true | false}`

Default: `true`

Data_type: bool, read/write

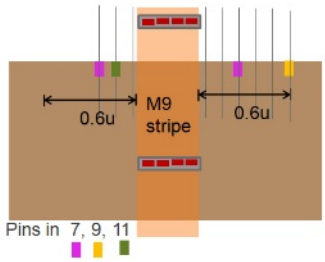
Controls abutted check for non-neighbour 2-pin and multi-partition pins.

Related Commands

- assign_io_pins
- assign_partition_pins
- align_pins
- check_pin_assignment
- edit_pin

Examples

- The following attribute prevents the generation of pins below power and ground strips:
  ```
  assign_pins_off_stripe below
  ```
- The following attribute will leave .6u of space between Metal9 stripe and Metal9 pins, and leave 1u of space between Metal11 stripe and Metal11 signal pin
  ```
  assign_pins_to_stripe_distance {{9 .6} {11 1}}
  ```

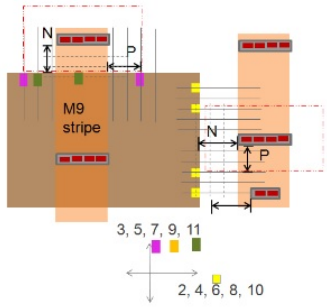


- For preferred direction pins, the following attributes will:
  - Leave .1u of space between Metal2,Metal4,Metal6 stack via to pins in same layer, and leave .3u of space between Metal9 stacked via and corresponding signal pin, on left/right edges
  - Leave .1u of space between Metal3,Metal5,Metal7 stack via to pins in same layer, and leave .3u of space between Metal9 stacked via and corresponding signal pin, on top/bottom edges

  ```
  assign_pins_to_via_distance_preferred_direction {{2 .1} {3 .1} {4 .1} {5 .1} {6 .1} {7 .1} {8 .3} {9 .3}}
  ```
- For any pins in non preferred direction, the following attributes will:
  - Leave .2u of space between Metal2,Metal4,Metal6 stack via to pins in same layer, and leave .4u of space between Metal9 stacked via and corresponding signal pin, on top/bottom edges
  - Leave .2u of space between Metal3,Metal5,Metal7 stack via to pins in same layer, and leave .4u of space between Metal9 stacked via and corresponding signal pin, on left/right edges

  ```
  assign_pins_pin_to_via_distance_non_preferred_direction {{2 .2} {3 .2} {4 .2} {5 .2} {6 .2} {7 .2} {8 .4} {9 .4}}
  ```

  Note: In the following picture, N stands for non-preferred and P stands for preferred.



change_current_design

```
change_current_design
[-help]
[partition_name]
```

Changes the partition view. It helps you to switch between different block cells and get back to the top cell.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `change_current_design` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man change_current_design`
`partition_name`	Specifies name of the partition to set as top cell. **Note:** • If no partition name is specified, then you are taken to the parent cell of the current top cell. • In nested partition designs, you can only switch to the next level partition. *Data_type*: string, optional

Examples

- The following commands changes sets `tdsp_top` as the top cell.
  `change_current_design tdsp_top`

Related Information

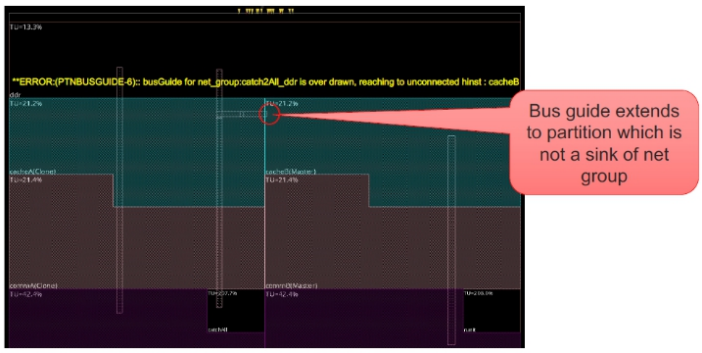
- Change Partition View chapter in the *Innovus Common UI Menu Reference*
- Partitioning the Design chapter in the *Innovus Stylus Common UI User Guide*

check_bus_guides

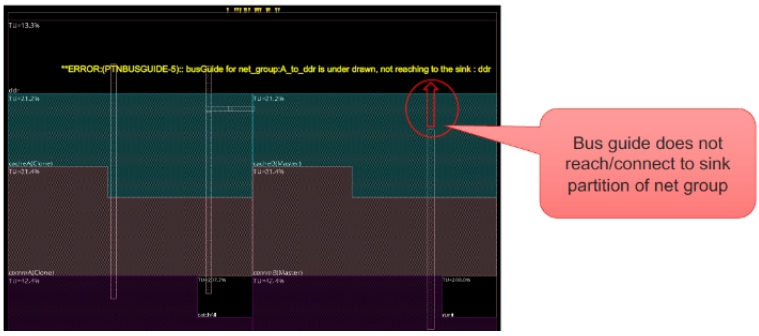
```
check_bus_guides
[-help]
```

Checks bus guides for violations during master-clone-aware bus guide operations. It checks for the consistency of net groups and bus guides by performing the following:

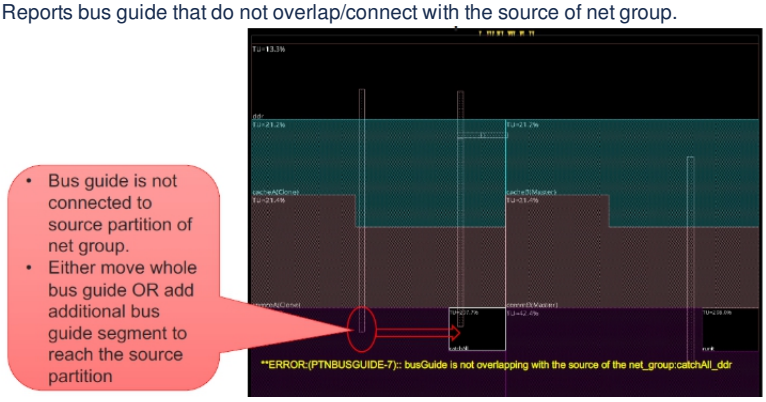
- Bus-guide physical checks:
  - Reports over-drawn bus guides.



  - Reports under-drawn bus guides (bus guide does not reach the sink of net group).



  - Reports bus guide that do not overlap/connect with the source of net group.



- Checks the net-group consistency and reports non-homogeneous nets in a net group.

Note: The `check_bus_guides` command does not support nested designs yet.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `check_bus_guides` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man check_bus_guides`

Example

- The following command checks the bus guides for violations.
  ```
  check_bus_guides
  ```

Related Information

- Partitioning the Design chapter in the *Innovus Stylus Common UI User Guide*

check_hier_route

```
check_hier_route
[-help]
[-partition {ptnName | ptnList}]
[-out_file outputFileName]
```

Checks and reports hierarchy violations for nets in the specified partitions.

The following conditions for hierarchy violations are checked:

- Intra-partition nets: All routes of this net should be within the partition boundary.
- Pure-Top nets: All routes of this net should be outside the partition boundary or touch the partition boundary from *outside*.
- Inter-partition nets: At least one route of this net must cross the partition boundary at partition pin. No route should cross the partition boundary at any other point.

Note: You can use this command to check for hierarchy violations before you push down the signal routes with the `commit_partitions -push_route` command.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `check_hier_route` parameter. For a detailed description of the command and all of its parameters, use the man command: `man check_hier_route`.
`-partition {ptnName | ptnList}`	Report the nets that violate the hierarchy with respect to the specified partition(s). *Default:* The command checks the hierarchy violations for nets on all partitions. *Data_type*: string, optional
`-out_file outputFileName`	Prints the details of net-wise violations in the specified file. *Default:* By default, the name of the file is `hier.rpt`. *Data_type*: string, optional

Examples of hierarchy violation conditions:

The following are examples of hierarchy violation conditions:

- If wires of a inter-partition net, are crossing the partition boundary at a place other than the corresponding pin shape.
- If wires of a inter-partition net, are crossing the partition boundary at multiple places.
- If wires of a pure top net, are crossing over a partition.
- If wires of a intra partition net are going outside the boundary of its partition.

Example

The following command checks the violations for nets on all the partitions in the design. A report of the violations is printed in the `hier.rpt` file.

```
check_hier_route
```

The output of this command on the display terminal is as follows:

```
------------------------------------------------------------
Summary of hierarchy violating nets with respect to each partition:
Partition: p1: 3 nets (0 intra, 2 inter, 1 top, 0 global)
Partition: p2: 1 nets (0 intra, 1 inter, 0 top, 0 global)
Partition: p3: 2 nets (1 intra, 0 inter, 1 top, 0 global)
Details of hierarchy violating nets reported in: hier.rpt
------------------------------------------------------------
```

The report printed in the `hier.rpt` file is as follows:

```
------------------------------------------------------------

Added the new header as follows:
############################################################
```

```
# Generated by: Cadence Innovus
# OS: Linux i686(Host ID icdlnx08s)
# Generated on:      Mon Mar 11 15:23:22 2015
#  Design: dtmf_chip
# Command: check_hier_route
############################################################
Partition: p1 . List of hierarchy violating nets against this partition:
Hierarchy violating net   Category   Hierarchy violating wire box(x1 y1 x2 y2)

--------------------------------------------------------------------------------
inter_ok_n1                 Inter-Partition "No wire crossing partition pin"
inter_hierarchy_violating_n Inter-Partition (2189.41 2549.82 2447.09 2550.10)
top_hierarchy_violating_n2  Top (1192.15 2310.14 1192.43 2695.14)

Partition: p2 . List of hierarchy violating nets against this partition:
Hierarchy violating net   Category   Hierarchy violating wire box(x1 y1 x2 y2)

--------------------------------------------------------------------------------
inter_ok_n1             Inter-Partition         "No wire crossing partition pin"
--------------------------------------------------------------------------------
```

Related Information

- commit_partitions
- Partitioning the Design chapter in the *Innovus Stylus Common UI User Guide*
  - Pushing Down Signal Routes

check_pin_assignment

```
check_pin_assignment
[-help] [-out_file file_name]
[-report_same_net_pin_as_overlap]
[-partitions partition_name list]
[-partitions partition_name list | {-pins_file file_name [-exclude_partitions ptn_name]} | -exclude_pin_file file_name]
[-report_violating_pin ]
[-ignore {bus_guide net_group pin_abutment pin_depth pin_group pin_guide pin_layer pin_min_area pin_on_fence pin_on_track pin_spacing
pin_width clones logical_pin pin_color unplaced} ]
[[-pre_check [-skip_check {overlap layer spacing_constraint fixed_io_pins multi_term_net master_clone net_group connection orient_relation
grid_track_alignment symmetry_offset master_clone_size align_zone}] [-exclude_net_file file_name] ] |
{ [-partitions partition_name list] [-pins {pin_name list}] [-pins_file file_name] [-exclude_pin_file file_name]
[-ignore {bus_guide net_group pin_abutment pin_depth pin_group pin_guide pin_layer pin_min_area pin_on_fence pin_on_track pin_spacing
pin_width clones logical_pin pin_color unplaced}]
[-report_violating_pin ]
[-exclude_partitions ptn_name]}]
```

Checks the generated partition pins and I/O pins for the following:

- Pins are assigned to the proper metal layer depending on the side of the partition
- Pins are assigned within the range of layers reserved for the partition
- Any particular location on a routing layer has only one pin assigned
- Pins are assigned on the routing track
- Pins have a valid spacing from:
  - routing blockages
  - power pins and ground pins
- Pins have the correct size, status, and type
- Pin assignment follows the minimum area rule
- If the `-minArea` parameter is specified, the command checks for and reports the pins that are:
  - floating from inside (that is, the pins that are connected only to the top level)
  - floating from both inside the partition and at the top level.
- Pins follow the *`keep out spacing`* constraints for a pin group or net group and reports any violations in pin spacing.
- Checks and reports pin violations on clones in a master clone design.

The summary report generated by the command also includes information about feedthrough pins. These reports also provide a summary for all pins of the partition, blackbox, and the top module. However, summary report for pins is only displayed when there are multiple objects. It does not provide such a summed up pin summary report when we use the `check_pin_assignment` command for individual objects. For example, when you use the `-partitions` option, the summary report for all pins is not displayed.

Notes:

- The `check_pin_assignment` command honors preferred routing layer attributes as soft constraint during pin assignment and reports violations in pin layers as an error.
- You can use this command after running early global routing and then partitioning with pin assignment.
- If this command is run on a blackbox that has one or more pins with a single LEF port but with multiple shapes within the LEF port, the command will check violations only with respect to the first shape and will ignore all other shapes. If this command is run on a blackbox that has one or more pins with multiple LEF ports, the command will check violations only with respect to the first LEF port and will ignore all other LEF ports.
- The `check_pin_assignment` command supports nested partitions and reports the pin status for pins of nested partitions.
- Each time the `check_pin_assignment` command is run, it automatically refreshes the violations as properties of pins. It clears all violation properties on all pins and populates fresh violation properties on each pin based on the latest `check_pin_assignment` results. These properties are persistent in save/restore DB. You can query results of the pin violations report generated by the `check_pin_assignment` command to automate the flow. For example:
  - To query for all partition pins with depth violation, you can use the following query:
    ```
    get_db [get_db partitions .master -if {.obj_type == hinst}] .local_hpins -if {.vio_depth == 1}
    ```
  - To query for all pins (partition pins, bbox pins and IO pins) with depth violation, you can use the following query:
    ```
    puts [concat \
    [get_db ports -if {.vio_depth == 1}] \
    ```

```
[get_db [get_db partitions .master -if {.obj_type == inst}] .pins -if {.vio_depth == 1}] \
[get_db [get_db partitions .master -if {.obj_type == hinst}] .local_hpins -if {.vio_depth == 1}] \
]
```

- To query for all pins in a master partition having pin depth violations, you can use the following query:
  ```
  get_db [get_db partitions .master -if {.obj_type == hinst && .name == "S"}] .local_hpins -if {.vio_depth == 1}
  ```

Parameters

`-help`	Outputs a brief description that includes type and default information for each `check_pin_assignment` parameter. For a detailed description of the command and all of its parameters, use the man command: `man check_pin_assignment`
`-exclude_partitions` *`ptn_name`*	Specifies the list of excluded partition for pin file. *Data_type*: string, optional
`-exclude_pin_file` *`file_name`*	Specifies pin-list to be excluded for partition in a file. *Data_type*: string, optional
`-exclude_net_file` *`file_name`*	Specifies the file containing the list of nets that are excluded while checking the pin assignment. *Data_type*: string, optional

`-ignore`	Ignores the mentioned list of categories from checking and reporting.
	• `bus_guide`: When specified the command ignores the violations of bus guide constraints.
	• `net_group`: When specified the command ignores the ordering and exclusion checks on nets.
	• `pin_abutment`:When specified the command ignores pin positions for abutted partitions.
	• `pin_depth`: When specified the command ignores violations of pin depth constraints.
	• `pin_group`: When specified the command ignores ordering and exclusion checks on pins.
	• `pin_guide`: When specified the command ignores violations of pin guide constraints.
	• `pin_layer`: When specified the command ignores violations of pin layer constraints.
	• `pin_min_area`:When specified the command ignores the pins that are:
	  ◦ floating from inside the partition
	  ◦ floating from both inside the partition and at the top level
	• `pin_on_fence`: When specified, the command ignores whether the pins are on fence or not.
	• `pin_on_track`: When specified the command ignores pins that are placed on layer tracks.
	• `pin_spacing`: When specified the command ignores violations of pin spacing constraints.
	• `pin_width`: When specified the command ignores violations of pin width constraints.
	• `clones`: When specified the command violations relating to clone pins.
	• `logical_pin`: When specified the command ignores violations of the logical pin.
	• `pin_color`: When specified the command ignores violations of pin color.
	• `unplaced`: When specified, the command ignores violations relating to unplaced pins.
	Data_type: (enum)+, optional

`-out_file` *`file_name`*	Specifies that in addition to displaying the report on screen, the command should also print the output in the specified file. *Default*: `<design_name>.checkPin.rpt` *Data_type*: string, optional

`-partitions` *partition_name list*	Specifies the partition for which the pins will be checked. This parameter is required if the partition is not the top-level module. *Data_type*: string, optional
`-pins {`*pin_name* `|` *pin_name_list*`}`	Specifies that the command should be run on the specified pin(s) only. *Data_type*: string, optional
`-pins_file` *file_name*	Specifies the pin list for a partition in a file. In the pin file, the pins and nets are specified in the following format: `Partition:` *PtnName/BlockName* *Pin1* *Pin2* .. *PinN* `Partition:` *PtnName/BlockName* *Pin1* *Pin2* .. *PinN* `Nets:` *netA* *netB* .. *netN* **Note:** While specifying the pin list in the pin_file, it is necessary to give the “Partition” keyword followed by the name of the partition or block. If this is not specified, no operations are performed. You can specify multiple partitions in the same pin_file. The pin_file does not support buses or wild cards for pin names. If a pin specified in the list or a partition is not present in the design, a warning is displayed. *Data_type*: string, optional
`-pre_check`	Checks for the feasibility of pin assignment mainly to catch the reason for unplaced pins in fully abutted design. **Note**: No other `check_pin_assignment` command parameter can be used if `-pre_check` is being used. *Data_type*: bool, optional
`-report_same_net_pin_as_overlap`	Reports the pins (connected to same net) placed at the same location and the same layer as overlap violation. *Data_type*: bool, optional
`-report_violating_pin`	Reports all the pins with violations. *Data_type*: bool, optional
`-skip_check {overlap layer spacing_constraint fixed_io_pins multi_term_net master_clone net_group connection orient_relation grid_track_alignment symmetry_offset master_clone_size align_zone}`	

	Skips checking the category from the list of type of checks. This parameter is used with the `-pre_check` parameter. • `overlap`: When specified, the command skips checking for any partition fences which are overlapping. • `layer`: When specified, the command skips checking for allowed layers on abutted edges of partition. • `spacing_constraint`: When specified, the command skips checking for spacing on abutted edges. • `fixed_io_pins`: When specified, the command skips checking if the block pin is placed on an abutted edge with its connected partition. • `multi_term_net`: When specified, the command skips checking for multi partition pin nets in abutted design. • `master_clone`: When specified, the command skips checking for master and clone partitions. • `net_group`: When specified, the command skips checking the net group ordering check in master and clone. • `connection`: When specified, the command skips checking if symmetric abutted designs have each master-clone pair connected to same pin. • `orient_relation`: When specified, the command skips checking the clone inside clone orientation. The clone inside a clone should have a relative orientation with its parent which is checked with master/clone inside master. • `grid_track_alignment`: When specified, the command skips checking for routing track offset in Master-Clone design to avoid off grid pins. • `symmetry_offset`: When specified, the command skips checking if the clone to clone offset (relativedistance on abutted edges) is same as in the master to master/clone connection pair. This is required so that the pins in each clone-clone and master-master/clone pair is the same. • `master_clone_size`: When specified, the command skips checking the master-clone size. This is required to check the number of poly edges of clones with master and check each edge size for any mismatch. • `align_zone`: When specified, the command skips checking for the common alignment zone between master-clone and their connected partitions. *Data_type*: (enum)+, optional

Example

- The following command checks violations on pins of partition `tdsp_core` and reports them to file `reportViolation.txt` :

  ```
  check_pin_assignment -partitions tdsp_core -out_file reportViolation.txt
  ```

  The output will look like:

  ```
  Start Checking pins of partition: [tdsp_core]
  Summary report for partition: [tdsp_core]
  Total Pins : 134
  Legally Assigned Pins : 110
  Illegally Assigned Pins : 0
  Unplaced Pins : 0
  Constant/Spl Net Pins : 24
  Internal Pins : 0
  Legally Assigned Feedthru Pins : 0
  Illegally Assigned Feedthru Pins : 0
  End of Summary report
  End Checking pins of partition: [tdsp_core]
  ```

- The following command accepts a file that contains the list of pins for partition A and partition B and checks any violations in pin assignment.

  ```
  check_pin_assignment -partitions {A B} -pins_file pinLst.txt
  ```

Related Information

- `assign_io_pins`
- `assign_partition_pins`
- `legalize_pins`
- Partitioning the Design chapter in the *Innovus Stylus Common UI User Guide*
  - Assigning Pins
  - Assigning I/O Pins
  - Validating Pin Placement
  - Working with Nested Partitions

clone_partition_obj

```
clone_partition_obj
[-help]
[-check_only]
[-objs {all place_blockage route_blockage pin_blockage macros fences bus_guides}]
[-redo]
[-ref_hinsts list_of_hinst_names]
[-sync]
[-target_hinsts list_of_hinst_names]
[-undo]
```

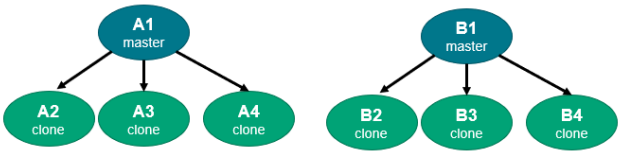
Syncs up the changes made to the specified objects in the master or clone partitions in the design with other hierarchical instances (master and clones) of the same module.

Parameters

`-check_only`	When specified, the tool check if the master and clone partitions are out of sync or not. *Data_type*: bool, optional
`-help`	Outputs a brief description that includes the type and default information for each `clone_partition_obj` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man clone_partition_obj`
`-objs {all place_blockage route_blockage pin_blockage macros fences bus_guides}`	When specified, the tool does the sync-up of the specified objects. The objects that can be synced up include place_blockage, route_blockage, pin_blockage, macros, fences, and bus_guides. *Data_type*: enum, optional
`-redo`	When specified, the changes made to the object are done again . The master and clone partitions are brought to the next committed sync state. **Note**: Currently this parameter only supports the `bus_guides` object type (-objs). **Note**: This parameter is exclusive with the `-ref_hinsts` and –`target_hinsts` parameter options. *Data_type*: bool, optional
`-ref_hinsts list_of_hinst_names`	Specifies the name of the master or clone hierarchical instance *from which* the obj data is to be copied. *Default*: All master partitions in the design. *Data_type*: string, optional
`-sync`	When specified, the objects in all master and clone partitions are synced up by looking at the changes in shape in any hierarchical instance of the master and clone partitions. **Note**: Currently this parameter only supports the `bus_guides` object type (-objs). **Note**: This parameter is exclusive with the `-ref_hinsts` and –`target_hinsts` parameter options. *Data_type*: bool, optional

`-target_hinsts` *list_of_hinst_names*	Specifies the name of the master or clone hierarchical instance *to which* the obj data is to be copied. • When a master partition is specified with the `-target_hinsts` parameter, specifying the `-ref_hinsts` becomes mandatory as a user provided hinst is considered to sync up objects. • When clones partitions are specified with the `-target_hinsts` parameter, specifying the `-ref_hinsts` is optional as master hierarchical instances are automatically considered (default) to sync-up objects *Default*: All clone partitions in the design. *Data_type*: string, optional
`-undo`	When specified, the changes made to the object are reverted. The master and clone partitions are brought to the last committed sync state. **Note**: Currently this parameter only supports the `bus_guides` object type (-objs). **Note**: This parameter is exclusive with the `-ref_hinsts` and –`target_hinsts` parameter options. *Data_type*: bool, optional

Example




```
Hinst: A1 (master), A2 (clone), A3(clone), A4 (clone)
Hinst: B1 (master), B2 (clone), B3(clone), B4 (clone)
```

- The following commands syncs all clones to their respective master.
  ```
  clone_partition_obj
  ```
- The following command syncs the hinsts A2 and A3 (clones) to the master of module A (A1).
  ```
  clone_partition_obj -target_hinsts {A2 A3}
  ```
- The following command syncs the hinsts A1 (master) and A3 (clone) to the hinst A2 (clone)
  ```
  clone_partition_obj -target_hinsts {A1 A3} -ref_hinsts {A2}
  ```
- The following command syncs the hinsts A1 (master), A3, B2, B3 (clones) to hinsts A2 (clone) and the master of module B (B1)
  ```
  clone_partition_obj -target_hinsts {A1 A3 B2 B3} -ref_hinsts {A2}
  ```
- The following command checks and reports if the hinsts A1(master) and A3, B2, B3 (clones) are synced to hinst A2 (clone) and the master of module B (B1)
  ```
  clone_partition_obj -target_hinsts {A1 A3 B2 B3} -ref_hinsts {A2} -check_only
  ```
- The following command checks and reports if the hinsts A1 (master) and A3, B2, B3 (clones) are synced to hinst A2 (clone) and the master of module B (B1) for bus guides only.
  ```
  clone_partition_obj -target_hinsts {A1 A3 B2 B3} -ref_hinsts {A2} -check_only -objs {bus_guides}
  ```
- The following command syncs the hinsts A1 (master) and A3, B2, B3 (clones) to hinst A2 (clone) and the master of module B (B1) for bus guides only.
  ```
  clone_partition_obj -target_hinsts {A1 A3 B2 B3} -ref_hinsts {A2} -objs {bus_guides}
  ```

Related Information

- `place_partition_clone`
- Partitioning the Design chapter in the *Innovus Stylus Common UI User Guide*

commit_partitions

```
commit_partitions
[-help]
[-add_pin_for_pg_wire_layers_in_distance {{layers} distance}]
[-input_nets_for_wire_distribution file_name]
[-no_top_pushdown]
[-no_via_cut_space]
[-partition partition_name...]
[-pg_pushdown_honor_1801]
[-pushdown_non_pg_special_net]
[-pushdown_selected_pg_bump]
[-push_route]
[-pushdown_honor_inside_pin_for_special_net]
[-pushdown_selected_pg_wires_as_route_blockage]
[-pushdown_special_net_as_obs {SNet list}]
[-skip_route_blockages_outside_partition]
[-tcf]
[-pin_cut_space ]
[-no_inherit_physical ]
```

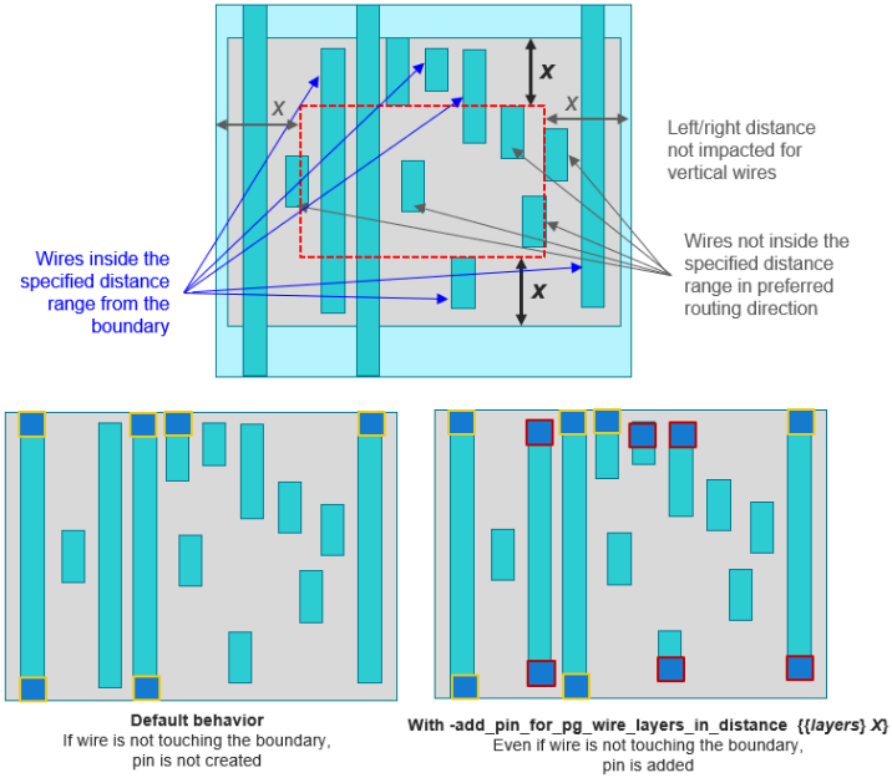
Converts the specified module fences to be partitions, pushes down the physical cells or power routing information to the partition level design(s), and duplicates strips in the partition that overlap with the partition boundary at their original widths. You can use this command after specifying the partition(s).

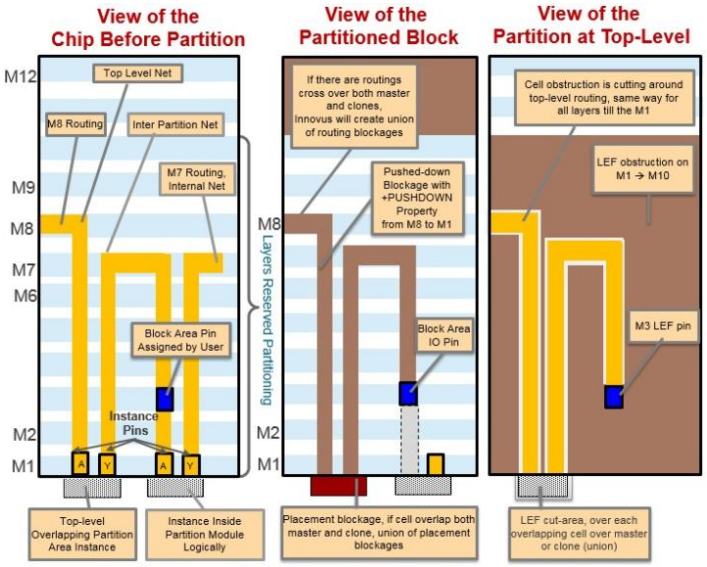
Notes:

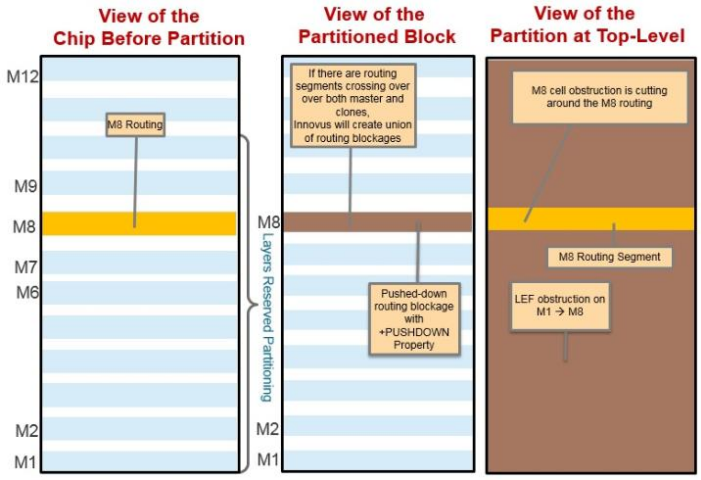
- While bringing back misaligned rows to top-level design, the `commit_partitions` command automatically detects mis-aligned rows and pushes down the row information as it is to the top-level design.
- To assign signal pins for the partitions, use the `assign_partition_pins` command.
- The `commit_partitions` command should be used in combination with the `assign_partition_pins` command. Cadence recommends not mixing these version 4.2 pin commands with previous (version 4.1 or earlier) pin commands in the design flow.
- While handling metal fills, the `commit_partitions` command removes the metal-fill shapes that cross the partition boundary. It pushes down only those fill-shapes that are inside the partition boundary and are on the metal-layers owned by the partition.
- In top-down flow, the partition command only promotes P/G pins from the list of pins promoted by the `set_promoted_macro_pin` command.
- The `commit_partitions` command supports partitions with flexmodels.
- The `commit_partitions` command pushes down the route blockage layer even when the layer type is master-slice. It creates routing blockages for master-slice layers, in the "OBS" section of LEF (from macro/IP/PADs), that are overlapping partition fences. With this feature:
  - The shape of the routing blockage is the same as the overlapping shape of the layer, in the "OBS" section.
  - The routing blockage is given a “pushdown” property, so that it is not brought back in `assemble_design`.
  - The union of overlapping shapes of layers in “OBS” section is taken for master and clones.
- The `commit_partitions` command automatically saves all the bus guide constraints with associated net groups. This enables bus routing at block level as per chip level constraints.
- The `commit_partitions` command automatically saves all the cell padding information to each partition block data. This information is saved in partitioned block .fp file.
- The `commit_partitions` command inherits all the attributes assigned to a net for block implementation flow. Preserving these attributes allows the Early Global Router and the NanoRoute Router to route the nets following specific requirements.
- The `commit_partitions` command now preserves all the pin shapes of nets.

Parameters

`-add_pin_for_pg_wire_layers_in_distance {{layers} distance}`

	Specifies the layers and distance to consider for creating pins on PG stripes that lie in the specified distance range from **Note:** Some designs shorten the PG stripes to avoid stripes of different net/voltages in different abutted partition abutting create any pins by default since the PG stripes do not touch the partition boundary. However, you can use the `-add_pin` wire shapes that lie in the specified distance range from the partition boundary. The `-add_pin_for_pg_wire_layers_in_` preferred direction of route) of the partition boundary to create pin shapes on them.  For example, the following command creates pins on PG stripes on layers M10-M13 that are within a distance of 5 um fr `commit_partitions -add_pin_for_pg_wire_layers_in_distance {{M10 M11 M12 M13} 5}` *Data_type:* string, optional
`-input_nets_for_wire_distribution` *file_name*	
	Specifies the file containing the net names to use for wire distribution. *Data_type:* string, optional
`-no_inherit_physical`	Specifies that physical only (filler) cells are not pushed down into the partition; they remain at the top level. *Data_type:* bool, optional
`-no_top_pushdown`	Specifies that top-level power routing is not pushed down from top-level design to partition blocks on layers not reserved *Data_type:* bool, optional
`-no_via_cut_space`	Specifies that there are no cuts in the obstructions around vias. You can use this parameter when you want to ensure tha corresponding cover blockages. *Data_type:* bool, optional

`-partition` *partition_name...*	Specifies the partition names to be committed. *Data_type:* string, optional
`-pg_pushdown_honor_1801`	When enabled, the UPF information is pushed down to partitions. *Data_type:* bool, optional
`-pin_cut_space`	Specifies that there are cuts in the obstructions around pins. *Default:* By default, there are no cuts in the obstructions around pins so that the top-level routing can connect the pins wi *Data_type:* bool, optional
`-push_route`	Pushes down signal routes when the design is partitioned. **Note:** All the wires of a hierarchy-violating net are discarded, in case even a single wire of that net has a hierarchy viola *Data_type*: bool, optional
`-pushdown_non_pg_special_net`	Specifies that the non PG special nets should also be pushed down into the partition. *Data_type:* bool, optional
`-pushdown_honor_inside_pin_for_special_net`	
	Preserves wire segments till partition pins at the top level while wire segments beyond partition pins are still pushdown a having a fixed status over the partition and all top- level nets crossing the partition boundary. The pure top level nets alo block data and the actual wire segments are retained at the top level. At the top-level: • All wires till the partition pin (with fixed status) are kept. For pure top nets, all wires are kept. • LEF obstructions are cut out for preserved wires segments. • LEF shape against overlapping top instances are cut out. At the block level: • Routing blockages till the partition pin for wires that were retained at top level are created. • All wires segments after the partition pin are retained. • Placement blockages are created for top instances overlapping partition fence area. **Note:** Before using this option, ensure that partition area pins – of the corresponding net – must be FIXED.  *Data_type:* bool, optional
`-pushdown_selected_pg_bump`	Specifies that only the selected PG bumps should be pushed down into the partition. The bumps which are not selected *Data_type:* bool, optional

`-pushdown_selected_pg_wires_as_route_blockage`	
	Pushes down selected PG routes as route blockages into the partition. This parameter enables you to select the wire se special PG Routes.  *Data_type:* bool, optional
`-pushdown_special_net_as_obs {SNet list}`	
	Specifies a list of special nets, which will be pushed down as routing blockages. It supports all types of special nets incl *Data_type:* string, optional
`-skip_route_blockages_outside_partition`	
	Skips pushing of objects which are close to partition fence. *Data_type:* bool, optional
`-tcf`	Enables tcf writing for the block. **Note:** A TCF file is used to store power related information of the nets in the design by keeping their toggle count relatec data of partition blocks. This data can be later saved using the `write_partitions` command into their respective directo *Data_type:* bool, optional

Example

- The following command converts fences to partition data and pushes down route crossing over fence `EUG` inside the partition boundary:
  ```
  commit_partitions EUG -push_route
  ```
- PG is Metal7, Metal8
  Partition is defined with reserved layer Metal6

  The following command copies PG from top to block for all the PG layers:
  ```
  commit_partitions -partition partitionName
  ```
  The following command cuts PG from top to block for PG layers reserved for partitions. For layers above the reserved layer, PG is not pushed down to ptn.
  ```
  commit_partitions -partition partitionName -no_top_pushdown
  ```
  If reserved layer is Metal7 then Metal7 PG is cut at top and pushed down and Metal8 PG will not be pushed down.

Related Information

- `assign_partition_pins`
- `assign_io_pins`
- `assemble_design`
- `flatten_partitions`
- Partitioning the Design chapter in the *Innovus Stylus Common UI User Guide*

commit_pushdown_eco

```
  commit_pushdown_eco
[-help]
-eco_obj_dir dir_name
-post_eco_dir dir_name
-pre_eco_dir dir_name
[-partition ptn_name_list]
```

Commits ECO for a partition block by applying ECO and generating a database for the block. It applies the previously generated ECO objects, to each partition, in parallel, using EDP (parallel/distributed) framework.

Note: The `commit_pushdown_eco` command also commits the FlexHtree amendment into the partition block. Clock amendments can be restored by using the `create_clock_tree_spec` command.

Parameters

`-eco_obj_dir dir_name`	Specifies the parent directory which contains all partition sub-directories in which ECO object files of partitions are kept. **Note:** The eco object file names are pre-determined to be: *eco_logical.txt* *eco_physical.txt* *eco_net.def.gz* *eco_pin.txt* *Data_type:* string, required
`-help`	Outputs a brief description that includes type and default information for each `commit_pushdown_eco` parameter. For a detailed description of the command and all of its parameters, use the man command: `man commit_pushdown_eco`.
`-partition ptn_name_list`	Specifies the list of partition names for which ECO objects are created. *Data_type:* string, optional
`-post_eco_dir dir_name`	Specifies the parent directory in which, after applying the ECOs, the partition databases are to be saved. *Default: ./post_eco_dir* *Data_type:* string, required
`-pre_eco_dir dir_name`	Specifies the latest version of original design, to which ECO objects need to be applied. It contains sub-directories to represent latest version of each partition database. *Data_type:* string, required

Related Information

- `create_pushdown_eco`
- `partition_push_network`
- Partitioning the Design chapter in the *Innovus Stylus Common UI User Guide*
  - Partition Pushdown Replay Flow

compare_pin_assignment_statistics

```
compare_pin_assignment_statistics
[-help]
[-ignore_min_wire_length min_wire_length]
[-out_file qor_compare_fine_name]
-reference reference_file_name
-target target_file_name
```

Generates a pin QoR comparison file based on a reference and a target pin QoR file.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `compare_pin_assignment_statistics` parameter. For a detailed description of the command and all of its parameters, use the man command: `man compare_pin_assignment_statistics`.
`-ignore_min_wire_length min_wire_length`	Specifies the minimum wire length. If reference wire length at a partition pin is less than the *`min_wire_length`*, then it is not reported in the output file. *Default*: 60 *Data_type*: string, optional
`-out_file qor_compare_fine_name`	Specifies an output file for the pin QoR comparison. *Default*: `./qor_compare.txt` *Data_type*: string, optional
`-reference reference_file_name`	Specifies the reference file generated using the `report_pin_assignment_statistics` command. For example, the following command generates a reference pin QoR file named `pin_qor_reference.txt`. `report_pin_assignment_statistics -outFile pin_qor_reference.txt` *Data_type*: string, required
`-target target_file_name`	Specifies the target file generated using the `report_pin_assignment_statistics` command. For example, the following command generates a target pin QoR file named `pin_qor_target.txt`. `report_pin_assignment_statistics -out_file pin_qor_target.txt` *Data_type*: string, required

Example

The following command generates a pin QoR comparison file, `pin_qor_compare.txt`:

```
report_pin_assignment_statistics -out_file pin_qor_reference.txt
report_pin_assignment_statistics -out_file pin_qor_target.txt
compare_pin_assignment_statistics -reference pin_qor_reference.txt -target pin_qor_target.txt -out_file pin_qor_compare.txt
```

Related Information

- `assign_partition_pins`
- `assign_io_pins`
- `report_pin_assignment_statistics`
- Partitioning the Design chapter in the *Innovus Stylus Common UI User Guide*

connect_macro_feedthru

```
connect_macro_feedthru
[-help]
[-nets string | -exclude_nets string | -selected_nets]
[-insts string | -exclude_insts string]
[-report_eco string | -check_only]
[-port_map string]
[-verbose]
[-max_search_distance float]
[-search_all_sides]
[-abutment float]
[-report_abutment string]
[-report_floating_ports string]
```

Connects predefined feedthrough pins in custom macros to overlapping nets that have wires crossing over the custom macros, based on the routing information. The feedthrough connectivity details are defined in a mapping file, which is specified through the `-port_map` *`filename`* parameter. You can include or exclude specific nets or instances. This command deletes routing only for the modified nets. You can run this command after placing the design and running Early Global Route.

Parameters

`-abutment` *`float`*	Specifies the distance that pins on the same track and layer must have from one another so as to be considered as abutted. If two or more pins are considered as abutted, any feedthrough path that passes through one of the pins will pass through the other pin(s) as well. *Default:* If this parameter is not specified, the default value is `1` micron. *Data_type*: float, optional
`-check_only`	Specifies that only a report of the messages should be displayed but no changes should be made to the netlist. *Data_type*: bool, optional
`-exclude_insts` *`string`*	Prevents feedthrough buffers from being added for the specified instance. The names of the instances are specified in a file, specified by *`fileName`*. In the file, you can specify wildcards (*) for the instance names to exclude. *Default:* All instances of the cell types specified in the portmap file are considered. *Data_type*: string, optional
`-exclude_nets` *`string`*	Prevents feedthrough buffers from being added to the specified net. The names of the nets are specified in a file, specified by *`fileName`*. In the file, you can specify wildcards (*) for the net names to exclude. *Default:* All nets or bus names are considered. *Data_type*: string, optional
`-help`	Outputs a brief description that includes the type and default information for each `connect_macro_feedthru` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man connect_macro_feedthru`
`-insts` *`string`*	Specifies that the feedthrough buffers should be added only for the specified instances. *Data_type*: string, optional
`-max_search_distance` *`float`*	Specifies the distance in microns from the wire crossing till which the command should search to find a connected feedthrough. **Note:** Specifying a large value for this parameter might have an adverse impact on the Quality of Results (QoR). *Data_type*: float, optional
`-nets` *`string`*	Specifies that the feedthrough buffers should be added only to the specified nets. *Data_type*: string, optional

`-port_map` *string*	Specifies the name of the file that contains the feedthrough connectivity definitions.
	Default: If you do not specify a file name, a file with the name `portmap` in the current directory is used.by default.
	Note: A file with the feethrough connectivity definitions must be available.
	The syntax of the file is as follows:
	`MACRO MacroName`
	`Macro definition section`
	`END MACRO`
	The definition of the macro is provided in the *`Macro definition`* section, which can contain one or more feedthrough sections. The name of the feedthrough section is optional.
	The syntax of the Feedthrough section is as follows:
	`Feedthrough [FeedthroughName]`
	`Pin Section`
	`END FEEDTHROUGH`
	Each Feedthrough section contains one section for the INPUT PIN and one section for the output pin.
	Note: Multi-fanout feedthrough sections are not supported.
	The syntax of the pin section is as follows:
	`PIN PinName`
	`DIRECTION {in | out };`
	`SIDE {left | right | top | bottom};`
	`END PIN`
	If the pin direction is not specified, it is read from the LEF file. If you specify a value for pin direction, ensure that it is consistent with the description in the LEF file.
	Note: The pin direction `inout` is not supported.
	Note: All the predefined macro feedthrough pins should be floating pins.
	Here is an example of a mapping file:
	`MACRO RAMXXX`
	`FEEDTHROUGH feedthrough1`
	`PIN feedthrough1_in`
	`END PIN`
	`PIN feedthrough1_out`
	`END PIN`
	`END FEEDTHROUGH`
	`FEEDTHROUGH feedthrough2`
	`PIN feedthrough2_in`
	`END PIN`
	`PIN feedthrough2_out`
	`END PIN`
	`END FEEDTHROUGH`
	`END MACRO`
	Data_type: string, optional

`-report_abutment` *string*	Reports, in the file specified, the list of pins that are considered as abutted. In the file, each feedthrough path is traced from the first input pin to the last input pin. Each path is preceded by the word `Begin` on a new line and followed by the word `End` on a new line.
	Here is the extract from an example file:
	`Begin`
	`Inst [Inst1], Term In [A1] Term Out [X1]`
	`Inst [Inst2], Term In [A2] Term Out [X2]`
	`End`
	In the previous example, the out term `X1` of instance `Inst1` is abutted with the in term `A2` of instance `Inst2`.
	Data_type: string, optional

`-report_eco` *string*	Specifies the name of the ECO file that will contain a report of the changes. You can use this file while performing ECO changes.
	Here is a sample report generated through this parameter:
	`FORMATVERSION 2`
	`ATTACHTERM eco_instance1 inputbuf2[2] testnet`
	`ADDNET FE_FTM_1_testnet`
	`ATTACHTERM eco_instance1 outputbuf2[2] FE_FTM_1_testnet ATTACHTERM i1 A FE_FTM_1_testnet ATTACHTERM eco_instance1 rstb FE_FTM_1_testnet`
	Data_type: string, optional

`-report_floating_ports` *string*	Specifies that the list of unused ports should be saved in the specified file. The file lists the instance name and the pin name for each unused port. After running the `connect_macro_feedthru` command, the unused ports should be manually tied. You can use this parameter to save the list of the unused ports to a file. *Data_type*: string, optional
`-search_all_sides`	Specifies that if the closest available feedthrough pin on the same side cannot be connected, the command should search for the feedthrough pins on other sides as well. When you specify this parameter, it is recommended to limit the distance of the search using the `-max_search_distance` option to get good QoR. *Default*: The command searches for a feedthrough whose in and out pins lie *on the same sides* of the macro on which the wires enter and exit the macro. *Data_type*: bool, optional
`-selected_nets`	Specifies that feedthrough buffers should be added only to the nets that are currently selected. *Default:* All nets.are considered. *Data_type*: bool, optional
`-verbose`	Provides additional messages, for example net-specific messages. *Data_type*: bool, optional

Example

The following command connects predefined feedthrough buffers in custom macros for nets that are listed in the file named `NetsToIncludeFile`. The command searches for a distance of 10 microns. The feedthrough connectivity is described in a file named `portmapFileForNets`.

```
connect_macro_feedthru portMap portmapFileForNets -nets NetsToIncludeFile -max_search_distance 10
```

Related Information

- `add_partition_feedthrus`
- Partitioning the Design chapter in the *Innovus Stylus Common UI User Guide*
  - Utilizing Pre-defined Feedthrough Pins in Custom Macros

convert_blackbox_to_fence

```
convert_blackbox_to_fence
[-help]
{-cell cellName | -inst instName}
```

Converts a blackbox to a fence. Once the black box netlist is well defined, use this command to convert the black box back to a regular partition.

Parameters

`-cell cellName`	Specifies the name of a module or submodule that should be converted to a fence. *Data_type*: string, optional
`-help`	Outputs a brief description that includes the type and default information for each `convert_blackbox_to_fence` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man convert_blackbox_to_fence`
`-inst instName`	Specifies the name of the blackbox instance that should be converted to a fence. *Data_type*: string, optional

Example

```
convert_blackbox_to_fence -inst hinst5
```

Related Information

- `create_blackbox`
- `convert_fence_to_black_box`
- Partitioning the Design chapter in the *Innovus Stylus Common UI User Guide*

convert_fence_to_black_box

```
convert_fence_to_black_box
[-help]
{-module cellName | -hinst hinstName}
[-size { x y }]
```

Converts a fence to a blackbox. Use this command before committing partition.

Parameters

`-module cellName`	Specifies the name of a module or submodule that should be converted to a black box. *Data_type*: string, optional
`-help`	Outputs a brief description that includes the type and default information for each `convert_fence_to_black_box` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man convert_fence_to_black_box`
`-hinst hinstName`	Specifies the name of the hierarchical instance that should be converted to a black box. *Data_type*: string, optional
`-size {x y}`	Specifies the width and height of the blackbox. *Default:* If you do not specify this parameter, the current fence size is taken as the initial size of the blackbox. *Data_type*: point, optional

Example

```
convert_fence_to_black_box -module cp1A -size 12 44
```

Related Information

- `create_blackbox`
- `convert_blackbox_to_fence`
- Partitioning the Design chapter in the *Innovus Stylus Common UI User Guide*

create_blackbox

```
create_blackbox
[-help]
[-core_spacing {left right top bottom}]
[-min_pitch_left integer]
[-min_pitch_right integer]
[-min_pitch_top integer]
[-min_pitch_bottom integer]
[-pin_layer_left {list_of_layers}]
[-pin_layer_right {list_of_layers}]
[-pin_layer_top {list_of_layers}]
[-pin_layer_bottom {list_of_layers}]
[-reserved_layer {list_of_layers}]
[-place_halo {left right top bottom}]
[-route_halo float]
[-route_halo_top_layer integer]
[-route_halo_bottom_layer integer]
[-skip_build_timing]
[ -cell string| -master_inst string]
{-size {x y}
    | -area float
    | -gate_count string | {-gate_area float [-cell_density float]
[ -include_macro_area {0 | 1}
{-macro_area float | -macro_list {macroName [count]}}]}}
[{-area float | -gate_count string}
[-min_width float | -min_height float | -fixed_width float | -fixed_height float ]]
[-min_width float | -min_height float | -fixed_width float | -fixed_height float |-aspect_ratio_range {min max}]
[-aspect_ratio float |-aspect_ratioRange {min max}]
```

Converts a hard macro into a blackbox. It is also used for specifying an instance to be the master of multiplying instantiated blackboxes. You can use this command after importing the design.

Note: You can convert a hard macro into a blackbox; however, Cadence does not recommend this because it will not update the blockage definitions when you change the blackbox size.

Note: For designs that have backside metal layers, both the backside and frontside layers are accepted. However, the automatic pin assignment flow is not impacted with the support for backside metal layers. It assigns pins on front layers only.

Parameters

`-area float`	Specifies the total area in square micrometers. If you specify the block size area, the default for the created block aspect ratio is 1. *Data_type*: float, optional
`-aspect_ratio float`	Specifies the aspect ratio value. This value combined with the area information determines the width and height. *Data_type*: float, optional
`-aspect_ratio_range {min max}`	Sets the minimum and maximum values for the aspect ratio range. This value combined with the specified related area information enables the blackbox to be reshaped while maintaining the current area. *Data_type*: point, optional
`-cell string`	Specifies the name of the cell. *Data_type*: string, optional
`-cell_density float`	Specifies the cell utilization. This value is used to derive the area of the blackbox, which is calculated using the following formula: `Blackbox area = (standard cell area / cell utilization) + macro area` *Default*: The default value of cell utilization is `1` (100%). *Data_type*: float, optional

`-core_spacing {left right top bottom}`	Specifies the space, in micrometers, between the module boundary and core design area of the module on the left, right, top, and bottom sides respectively. The partition pins will be located at the partition instance boundary and the core area. *Data_type*: string, optional
`-fixed_height float`	Specifies the fixed height of the module. *Data_type*: float, optional
`-fixed_width float`	Specifies the fixed width of the module. *Data_type*: float, optional
`-gate_area float`	Specifies the gate area. If the value of the `-include_macro_area` parameter is `0`, the gate area is equal to the standard cell area. If the value of the `-include_macro_area` parameter is `1`, the gate area also includes the macro area. *Data_type*: float, optional
`-gate_count string`	Specifies a gate count value. The total cell area is equal to the specified gate count multiplied by the specified area per gate. • *count:* Specifies the gate count value. • *areaPerGate:* Specifies an area per gate value in square micrometers. To get the cell area, this value is multiplied by the specified gate count. • *utilizationValue:* Specifies the cell utilization. This value derives the area that is reserved for routing. The final block size is equal to the total cell area plus the routing area. *Default*: The default value is 0.7 (70%) *Data_type*: string, optional
`-help`	Outputs a brief description that includes type and default information for each `create_blackbox` parameter. For a detailed description of the command and all of its parameters, use the man command: `man create_blackbox`
`-include_macro_area {0 | 1}`	A value of `0` specifies that the gate area does not include the macro area specified through the `-macro_area` or the `-macro_list` parameter. A value of `1` specifies that the gate area includes the macro area specified through the `-macro_area` or the `-macro_list` parameters. *Default:* The default value is `0`. *Data_type*: enum, optional

`-macro_area float`	Specifies the total area of the macro(s) in square microns.
	• If the value of the `-include_macro_area` parameter is set to 0:
	`Total blackbox area = (gate area / cell utilization value) + macro area`
	• If the value of the `-include_macro_area` parameter is set to 1:
	`Total blackbox area = ((gate area - macro area) / cell utilization value) + macro area`
	Data_type: float, optional

`-macro_list {macroName [count]}`	Specifies the name of the hard macro whose area will be included while calculating the total gate area. The `macroName` argument specifies the name of the macro, and the `count` argument specifies the number of times the macro is referenced. Thus, by specifying a value for count, you avoid repeating the macro names for macros that are referenced multiple times.
	• If the value of the `-include_macro_area` parameter is set to 0:
	`Total blackbox area = (gate area / cell utilization value) + macro area`
	• If the value of the `-include_macro_area` parameter is set to 1:
	`Total blackbox area = ((gate area - macro area) / cell utilization value) + macro area`
	Default: The default value of `count` is `1`.
	Data_type: string, optional

`-master_inst string`	Specifies the name of the master blackbox. *Data_type*: string, optional
`-min_height float`	Specifies the minimum height of the specified module. *Data_type*: float, optional

`-min_pitch_left integer`	Specifies the pin pitch dimension for the *Left* side of the blackbox. The default is 2, which places one pin for every two metal tracks. The actual pin pitch depends on the pin's metal layer. *Data_type*: int, optional
`-min_pitch_right integer`	Specifies the pin pitch dimension for the *Right* side of the blackbox. The default is 2, which places one pin for every two metal tracks. The actual pin pitch depends on the pin's metal layer. *Data_type*: int, optional
`-min_pitch_top integer`	Specifies the pin pitch dimension for the *Top* side of the blackbox. The default is 2, which places one pin for every two metal tracks. The actual pin pitch depends on the pin's metal layer. *Data_type*: int, optional
`-min_pitch_bottom integer`	Specifies the pin pitch dimension for the *Bottom* side of the blackbox. The default is 2, which places one pin for every two metal tracks. The actual pin pitch depends on the pin's metal layer. *Data_type*: int, optional
`-min_width float`	Specifies the minimum height of the specified module. *Data_type*: float, optional
`-pin_layer_left {list_of_layers} -pin_layer_right {list_of_layers} -pin_layer_top {list_of_layers} -pin_layer_bottom {list_of_layers}`	Specifies the metal layers to use for the blackbox. The metal layers correspond to the sides of the blackbox, where the vertical metal layers (for example, *Metal2, Metal4*, and so on) are for the Top and Bottom sides, and horizontal metal layers (for example, `Metal3, Metal5`, and so on) are for the Left and Right sides of the blackbox. Specify this value in integers, for example `-pin_layer_top {2 4}` *Data_type*: string, optional
`-place_halo {left right top bottom}`	Specifies extra spacing, in micrometers around the sides (left, bottom, right and top respectively) of the blackbox that should not be used for placement. At the top-level design, this information is saved as part of the partition section in a floorplan file. This information is also saved in a partition floorplan file when saving partitions. By default, the value is 0 for all the sides. *Data_type*: string, optional
`-reserved_layer {list_of_layers}`	Specifies the metal layers that are used for routing in the partition and generating partition pins. Any metal layers that are not specified, usually the top-most metal layers, are allowed to route over the partition.A normal six-metal layer specification is Metal1, Metal2, Metal3, Metal4 and Metal5 specified, and Metal6 unspecified. When saving the partition, the LEF generated for this partition will have routing blockages on their layers so that the top-level router is aware of which metal layers are being used in the partition. *Data_type*: string, optional
`-route_halo float`	Specifies routing spacing, in micrometers, around the sides of the blackbox. You can specify a positive or a negative value for the routing halo. A positive value means that the halo will be outside the blackbox. A negative value means that the halo will be applicable during block implementation, inside of the blackbox. The same value is used for all sides of the blackbox. *Data_type*: float, optional
`-route_halo_bottom_layer integer`	Specifies the bottom partition layer for which routing halo will be created. *Default*: By default, the bottom layer reserved for the partition is used. *Data_type*: int, optional
`-route_halo_top_layer integer`	Specifies the top partition layer for which routing halo will be created. *Default*: By default, the top layer reserved for the partition is used. *Data_type*: int, optional
`-size x y`	Specifies fixed width and height of a specified module, in micrometers. *Data_type*: point, optional

`-skip_build_timing`	Skips re-building the time graph and re-initializing the top cell information in order to improve the run time of the blackbox specification for designs with a large number of blackboxes. **Note:** Since the timing graph only needs to build for the last blackbox, you must specify the final `create_blackbox` command without this parameter. For example, if your design has three blackboxes, the first two `create_blackbox` commands should be specified with the `-skip_build_timing` parameter while the last `create_blackbox` command should be specified without this parameter. *Data_type*: bool, optional

Examples

- The following command converts the hard macro `hn246` into a black box. The gate area is specified as 100000 square microns and the specified macro area is 220 square microns. The cell utilization value is not specified, and so the default value of 1 is used. The `-include_macro_area` parameter is not specified and so the default value of 0 is used.
  When the `-include_macro_area` parameter is 0, the blackbox area is calculated using the formula:
  `Total blackbox area = (gate area / cell utilization value) + macro area`
  The total black box area in this case is, therefore, (100000 / 1) + 220 that is 100220 square microns.
  ```
  create_blackbox -cell hn246 -gate_area 100000 -macro_area 220
  ```
- The following command is similar to the command in the previous example, except that the value of `-include_macro_area parameter` is set to 1 and the value of the `-cell_density` parameter is set to `.7`
  When the `-include_macro_area` parameter is set to 1, the blackbox area is calculated using the formula
  `Total blackbox area = ((gate area - macro area)/cell utilization value) + macro area`
  The total blackbox area in this case is, therefore, ((100000 -220) /.7 + 220) that is 142762.86 square microns.
  ```
  create_blackbox -cell hn246 -gate_area 100000 -include_macro_area 1 -macro_area 220 -cell_density 0.7
  ```
- The following command converts the macro `hn246` into a blackbox and specifies that the areas of macros `macro_2A` and `macro_2B` should be added while calculating the blackbox area. The macro named `macro_2A` is included four times and the macro named `macro_2B` is included three times.
  ```
  create_blackbox -cell hn246 -gate_area 100000 -macro_list macro_2A 4 macro_2B 3
  ```

Related Information

- `convert_blackbox_to_fence`
- `convert_fence_to_black_box`
- `delete_black_boxes`
- `get_black_box_area`
- `read_black_box_netlist`
- `set_blackbox_master_orient`
- `set_blackbox_master_to_r0`
- `write_black_box_lef`
- Partitioning the Design chapter in the *Innovus Stylus Common UI User Guide*
  - Defining Blackboxes

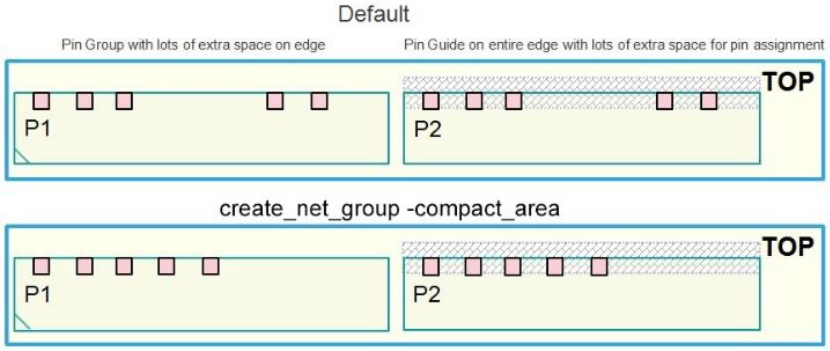
create_net_group

```
  create_net_group
[-help]
[-mixed_signal]
[-spacing minSpaceInTracks]
[ {[-opt_order ]} | -spread_pins | -compact_area ]
[-exclude_pins {all_layer | same_layer | all_layer_in_guided_area}
[-keep_out_spacing integer]]
{ [-name netGroupName [-nets netNameList]] }
```

Creates an empty net group. A net group can contain net names, bus names, or segments of a bus. To add nets into the net group, use the `update_net_group` command. You can use this command after importing the design.

Note: If a net is part of a net group, and you add the same net to another net group, the Innovus software will display a warning, but the net will be added to the second net group too. However, when pin assignment is performed, the net is considered part only of the *first* net group to which it was added – and *not* part of any other net group to which it was subsequently added.

Parameters

`-compact_area`	Specifies that pins in the group are assigned tightly together. By default, pins which are part of a group and associated to a guide can be spread inside the guide based on the area available and alignment to targets. The `compact_area` option, when used, ignores the alignment and available area and forces the pins to the minimum area on guide honoring the minimum spacing constraint and overlooking any alignment to targets.  *Data_type*: bool, optional
`-exclude_pins {all_layer | same_layer | all_layer_in_guided_area}`	
	Specifies that pins in the group exclude foreign pins on all layers, same layer, or on all layers in the guided area. • `all_layer`: Specifies that pins in the group exclude foreign pins between grouped pins on all layers. • `same_layer`: Specifies that pins in the group exclude foreign pins grouped pins on all same layer. • `all_layer_in_guided_area`: Specifies that pins in the group exclude foreign pins in the guide area defined for the group, on all layers. **Note:** If guide is not defined for a group down the flow, `all_layer_in_guided_area` option would be treated like `-exclude_pins {all_layer}` for the group without guides behavior. *Data_type*: enum, optional
`-help`	Outputs a brief description that includes type and default information for each `create_net_group` parameter. For a detailed description of the command and all of its parameters, use the man command: `man create_net_group`.
`-keep_out_spacing integer`	Specifies the minimum spacing between the pins placed outside of the group, in number of tracks or pitch. *Minimum:* 0 *Maximum:* 65535 *Data_type*: int, optional

`-mixed_signal`	Specifies to identify a mixed signal group. *Data_type*: bool, optional
`-nets` *netNameList*	Specifies the name of the net(s). You can use wildcards (`*?`) with this parameter. *Data_type*: string, optional
-name netGroupName	Specifies the name of the created net group name. *Data_type*: string, optional
`-opt_order`	Specifies that pins in the group are reordered to optimize wire length. If this option is not selected, the pin order is exactly as specified in the pin group. *Data_type*: bool, optional
`-spacing` *minSpaceInTracks*	Specifies the minimum spacing between the pins of the net group. *Default:* The default value is two tracks. *Data_type*: int, optional
`-spread_pins`	Specifies that pins in the group must be distributed evenly. *Data_type*: bool, optional

Example

- The following command creates a net group `bundle1`:
  ```
  create_net_group bundle1
  ```
- The following command creates a net group `bundle1` and allows re-ordering of the nets definition inside the group:
  ```
  create_net_group bundle1 -opt_order
  ```

Related Information

- `update_net_group`
- `delete_net_group`
- `delete_nets_from_net_group`
- `report_net_groups`
- Floorplanning the Design chapter in the *Innovus Stylus Common UI User Guide*
- Partitioning the Design chapter in the *Innovus Stylus Common UI User Guide*
  - Setting Pin Constraints
  - Net Group

create_partition

```
create_partition
[-help]
[-core_spacing {left right top bottom}]
[-min_pitch_bottom x]
[-min_pitch_left x]
[-min_pitch_right x]
[-min_pitch_top x]
[-pin_layer_left {list_of_layers}]
[-pin_layer_right {list_of_layers}]
[-pin_layer_top {list_of_layers}]
[-pin_layer_bottom {list_of_layers}]
[-place_halo {left right top bottom}]
[-rail_width railWidth]
[-reserved_layer {list_of_layers}]
[-route_halo floatValue]
[-route_halo_bottom_layer integerValue]
[-route_halo_top_layer integerValue]
[-std_cell_height x]
{-hinst hinst_name | -proto_hinsts {master_hinst_name_list} | -cell cell_name}
[-check_clone_alignment]
[-hinst hinst_name  [-copy_from ref_partition_name]] [-cell cell_name [-copy_from ref_partition_name]]
```

Defines a new partition corresponding to a hierarchical instance. The `create_partition` command supports nested partitions and can be used to specify partitions inside a partition. While specifying partitions, you can specify the definition in any order. You can use this command after importing the design.

Note: The `create_partition` command does not check for fence constraint on clones and does not align them to the partition with respect to the master. If the clones do not have fence constraint, `create_partition` makes fences for clone modules.

Note: For designs that have backside metal layers, both the backside and frontside layers are accepted. However, the automatic pin assignment flow is not impacted with the support for backside metal layers. It assigns pins on front layers only.

Parameters

`-cell cell_name`	Specifies the name of the cell for defining a new partition at an instance that has R0 orientation. **Note**: For black-boxes, an instance with non-R0 orientation is chosen if an instance with R0 orientation is not found. *Data_type*: string, optional
`-check_clone_alignment`	Specifies to check only the alignment of the clone partition with the master partition. *Data_type*: bool, optional
`-copy_from ref_partition_name`	Specifies the name of the reference partition from which to copy their properties. While copying the constraints, the shape, orientation, and location are considered: • **Orientation**: The orientation of the partition being defined is copied from the reference partition (*ref_partition_name*). If the reference partition has clones, then the orientation of the master is considered. • **Shape**: The shape of the partition being defined is copied from the reference partition (*ref_partition_name*). • **Location**: The fence location of the partition hinst being defined takes priority. In case the target partition hinst does not have a fence constraint (location), the location information is copied from the reference partition master hinst. This will result in overlapping fences. **Note:** If the `-copy_from` parameter is being used with the -cell parameter that has multiple Hinsts, then the location of the hinst that has a fence (location) constraint is considered as the master of the partition being defined. *Data_type*: string, optional
`-core_spacing {left right top bottom}`	

	Specifies the space, in micrometers, between the module boundary and core design area of the partition module on the left, right, top, and bottom sides respectively. The partition pins will be located at the partition instance boundary and the core area. **Note**: These core-to-edge values are always synchronized between the partition and the corresponding power domain as follows: If the specified instance was also earlier defined as a power domain, these values for the partition, by default, will have the same value as the values for the power domain. If you specify different values for the partition, the values for the power domain are updated so that they are the same as the values defined for the partition. Similarly, if a power domain is created (or an existing power domain modified) on the specified instance after the partition is defined, the values set earlier for the partition are updated so that they are the same as the values defined for the power domain. *Data_type*: string, optional
`-help`	Outputs a brief description that includes the type and default information for each `create_partition` parameter. For a detailed description of the command and all of its parameters, use the man command: `man create_partition`
`-hinst` *hinst_name*	Specifies the name of the hierarchical instance. **Note**: None of its ancestors or descendants in the netlist tree can be a partition. *Data_type*: string, optional
`-min_pitch_bottom` *x*	Specifies the pin pitch dimension for the bottom partition side. The default is 2, which places one pin for every two metal tracks. The actual pin pitch depends on the pin's metal layer. The Partition Pin Guide floorplan object will supersede this entry and use the Min. Space entry in the object's Attribute Editor. *Data_type*: int, optional
`-min_pitch_left` *x*	Specifies the pin pitch dimension for the left partition side. The default is 2, which places one pin for every two metal tracks. The actual pin pitch depends on the pin's metal layer. The Partition Pin Guide floorplan object will supersede this entry and use the Min. Space entry in the object's Attribute Editor. *Data_type*: int, optional
`-min_pitch_right` *x*	Specifies the pin pitch dimension for the right partition side. The default is 2, which places one pin for every two metal tracks. The actual pin pitch depends on the pin's metal layer. The Partition Pin Guide floorplan object will supersede this entry and use the Min. Space entry in the object's Attribute Editor. *Data_type*: int, optional
`-min_pitch_top` *x*	Specifies the pin pitch dimension for the top partition side. The default is 2, which places one pin for every two metal tracks. The actual pin pitch depends on the pin's metal layer. The Partition Pin Guide floorplan object will supersede this entry and use the Min. Space entry in the object's Attribute Editor. *Data_type*: int, optional
`-pin_layer_bottom {`*list_of_layers*`}`	Specifies the metal layers to use for the partition. The metal layers correspond to the sides of the partition, where the vertical metal layers (for example, Metal2, Metal4, and so on) are for the Top and Bottom sides, and horizontal metal layers (for example, Metal3, Metal5, and so on) are for the Left and Right sides of the partition. Specify this value in integers, for example `-pin_layer_bottom {2 4}` **Note**: Only layers in their preferred routing direction are accepted. **Note**: If these option is not supplied, all allowed layers in preferred routing direction will be assumed for that side. **Note:** For designs that have backside metal layers, both the backside and frontside layers are accepted. *Default*: All allowed layers in preferred direction. *Data_type*: string, optional

`-pin_layer_left {list_of_layers}`	Specifies the metal layers to use for the partition. The metal layers correspond to the sides of the partition, where the vertical metal layers (for example, Metal2, Metal4, and so on) are for the Top and Bottom sides, and horizontal metal layers (for example, Metal3, Metal5, and so on) are for the Left and Right sides of the partition. Specify this value in integers, for example `-pin_layer_left {3 5}` **Note**: Only layers in their preferred routing direction are accepted. **Note**: If these option is not supplied, all allowed layers in preferred routing direction will be assumed for that side. **Note:** For designs that have backside metal layers, both the backside and frontside layers are accepted. *Default*: All allowed layers in preferred direction. *Data_type*: string, optional
`-pin_layer_right {list_of_layers}`	Specifies the metal layers to use for the partition. The metal layers correspond to the sides of the partition, where the vertical metal layers (for example, Metal2, Metal4, and so on) are for the Top and Bottom sides, and horizontal metal layers (for example, Metal3, Metal5, and so on) are for the Left and Right sides of the partition. Specify this value in integers, for example `-pin_layer_right {3 5}` **Note**: Only layers in their preferred routing direction are accepted. **Note**: If these option is not supplied, all allowed layers in preferred routing direction will be assumed for that side. **Note:** For designs that have backside metal layers, both the backside and frontside layers are accepted. *Default*: All allowed layers in preferred direction. *Data_type*: string, optional
`-pin_layer_top {list_of_layers}`	Specifies the metal layers to use for the partition. The metal layers correspond to the sides of the partition, where the vertical metal layers (for example, Metal2, Metal4, and so on) are for the Top and Bottom sides, and horizontal metal layers (for example, Metal3, Metal5, and so on) are for the Left and Right sides of the partition. Specify this value in integers, for example `-pin_layer_top {2 4}` **Note**: Only layers in their preferred routing direction are accepted. **Note**: If these option is not supplied, all allowed layers in preferred routing direction will be assumed for that side. **Note:** For designs that have backside metal layers, both the backside and frontside layers are accepted. *Default*: All allowed layers in preferred direction. *Data_type*: string, optional
`-place_halo {left right top bottom}`	
	Specifies extra spacing, in micrometers (around the left, bottom, right and top sides respectively) of the partition that should not be used for placement. The placement program will not place standard cells in the area for partition placement. This spacing is called placement halo. At the top-level design, this information is saved as part of the partition section in a floorplan file. This information is also saved in a partition floorplan file when saving partitions. By default, the value is 0 for all the sides. If the hierarchical instance was earlier defined as a power domain, then this field corresponds to the minimum gap parameter of the power domain. **Note**: The minimum gap value of the power domain and the placement halo values of the partition are always synchronized as follows: If the specified instance was also earlier defined as a power domain, the placement halo values for the partition, by default, will have the same value as the minimum gap values for the power domain. If you specify a different value for the partition, the value for the power domain is updated so that it is the same as the value defined for the partition. Similarly, if a power domain is created (or an existing power domain modified) on the specified instance after the partition is defined, the value set earlier for the partition is updated so that it is the same as the value defined for the power domain. **Note**: The placement halo created by the `create_partition` command cannot be honored by `place_design` before partitions have not been committed. *Data_type*: string, optional
`-proto_hinsts {master_hinst_name_list}`	
	Specifies the master hierarchical instances list for the prototyping flow. It defines new partitions for the specified hierarchical instances with guide constraints. It is used before `proto_design`, which will convert the physical constraint from guide to fence finally. *Data_type*: string, optional

`-rail_width railWidth`	Specifies the standard cell power rail width, in micrometers. If partition pins are not allowed above power or ground rail width, specify the dimension of the power rail width. If a value of 0.0 is entered (the default), the width from the Technology file will be used. *Data_type*: float, optional
`-reserved_layer {list_of_layers}`	
	Specifies the metal layers that are used for routing in the partition and generating partition pins. Any metal layers that are not specified, usually the top-most metal layers, are allowed to route over the partition. A normal six-metal layer specification is Metal1, Metal2, Metal3, Metal4 and Metal5 specified, and Metal6 unspecified. When saving the partition, the LEF generated for this partition will have routing blockages on their layers so that the top-level router is aware of which metal layers are being used in the partition. **Note:** For designs that have backside metal layers, both the backside and frontside layers are accepted. *Data_type*: string, optional
`-route_halo floatValue`	
	Specifies routing spacing, in micrometers, around the sides of the partition. A routing halo is honored only by the signal router. The signal router treats routing on the specific routing layers in the routing halo area as very high cost routing. However, perpendicular routing or straight connections to pins are acceptable. The special router does not honor routing halos. To block special routing, use routing blockages instead. You can specify a positive or a negative value for the routing halo. A positive value means that the halo will be outside the partition. A negative value means that the halo will be applicable during block implementation, inside of the partition. The same value is used for all sides of the partition. **Note**: For instance blocks - blackboxes, hard macros or block-level designs - specify the routing halo through the *Floorplan - Edit Floorplan - Edit Halos* form in the GUI or through the `create_route_halo` command. Also, for bottom- up hierarchical flow, specify routing halo for a block at the top-level design or at the block-level design through the *Floorplan - Edit Floorplan - Edit Halos* form in the GUI or through the `create_route_halo` command. *Data_type*: float, optional
`-route_halo_top_layer integerValue`	
	Specifies the top partition layer for which routing halo will be created. *Default*: By default, the top layer reserved for the partition is used. *Data_type*: int, optional
`-route_halo_bottom_layer integerValue`	
	Specifies the bottom partition layer for which routing halo will be created. *Default*: By default, the bottom layer reserved for the partition is used. *Data_type*: int, optional
`-std_cell_height x`	Specifies the standard cell height for the partition. *Default*: The default value is derived from the default technology site and row height. *Data_type*: float, optional

Example

```
create_partition -hinst ctr_inst
-core_spacing 0.56 0.56 0.0 0.0
-rail_width 0.0
-min_pitch_left 2
-min_pitch_right 2
-min_pitch_top 2
-min_pitch_bottom 2
-reserved_layer {1 2 3 4}
-pin_layer_top {2 4}
-pin_layer_left {3}
-pin_layer_bottom {2 4}
-pin_layer_right {3}
-place_halo 10.0 0.0 0.0 0.0
-route_halo 10.0
-route_halo_top_layer 7
-route_halo_bottom_layer 1
```

Related Information

- commit_partitions
- delete_partitions
- write_partitions
- Partitioning the Design chapter in the *Innovus Stylus Common UI User Guide*
  - Defining Partitions

create_partition_feedthru

```
create_partition_feedthru
[-help]
-area x1 y1 x2 y2
[-layer]
[-name string]
```

Creates a partition routing feedthrough object. The object area keeps pins from being created on the specified metal layer. A cdump file with routing obstruction is created for the top level partition (done when saving the partition). You can use this command during partition floorplanning.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `create_partition_feedthru` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man create_partition_feedthru`
`-area x1 y1 x2 y2`	Specifies the coordinates (*x1 y1 x2 y2)* of the feedthrough area. *Data_type*: rect, optional
`-layer`	Specifies the metal layer where pins are not to be created. **Note:** If `-layer` is not specified then by default feed-through is created on all the partition layers. *Data_type*: int, optional
`-name string`	Specifies the name of the routing feedthrough. *Data_type*: string, optional

Example

The following command creates a feedthrough object to block pins being generated on layer *Metal5*:

```
create_partition_feedthru 2353.8000 7123.000 2653.8000 8298.2000 5
```

Related Information

- `add_partition_feedthrus`
- `delete_all_partition_feedthrus`
- Partitioning the Design chapter in the *Innovus Stylus Common UI User Guide*
  - Inserting Feedthroughs

create_partition_wrapper

```
create_partition_wrapper
[-help]
-partitions partition_names
[-pushdown]
[-wrapper_name wrapper_suffix_string]
```

Creates a wrapper partition that embeds the specified partition cell to preserve its verilog module definition in the netlist. The wrapper partition can be used for making netlist changes without changing the netlist of the embedded (wrapped) cell definition. The wrapper partition has the same partition definition attributes as the specified partition. The timing constraints (through SDC) and low power information (through UPF) of the wrapper partition is updated with the correct hierarchical pin information. The pin constraints applied on pins of the original partition are mapped to the new wrapper pins.

Note: After the wrapper partition is created using the `create_partition_wrapper` command:

- the original partition ceases to exist (`delete_partitions`).
- the fence attributes and pin constraints of the original partition are removed.
- the module pins of the original partition as preserved such that they cannot be altered later.

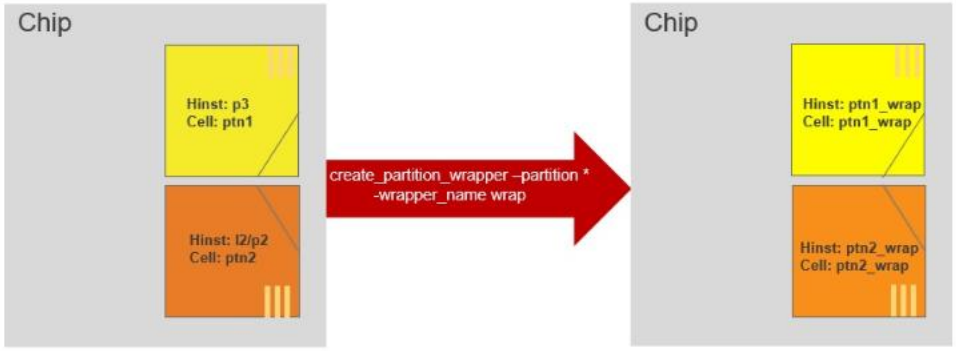
Parameters

`-help`	Outputs a brief description that includes the type and default information for each `create_partition_wrapper` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man create_partition_wrapper`
`-partitions partition_names`	Specifies the name of one or more partitions on which the wrapper is created. **Note:** The partition_wrapper will have the same: • Number, name and locations (If already assigned) of pins • Area, shape, location, orientation *Data_type*: string, optional
`-pushdown`	When enabled, it keeps the partition hinst intact for the wrapper partition. *Data_type*: bool, optional
`-wrapper_name wrapper_suffix_string`	Specifies the suffix that is added to a partition name for creating a partition wrapper. *Default*: <`ptnName>_wrapper` *Data_type*: string, optional

Example

The following command creates a wrapper partition with suffix wrap for all the partitions in the design:

```
create_partition_wrapper -partition * -wrapper_name wrap
```



Related Information

- `delete_partitions`
- Partitioning the Design chapter in the *Innovus Stylus Common UI User Guide*

create_pin_blockage

```
create_pin_blockage
[-help] [-cell cell_name] [-layer {layer_id_list}]
{ -area {x1 y1 x2 y2} | [-edge edge_number [[-offset_start distance] [-offset_end distance]]] |
[-abutted_master_clone_channel_access_only [-feedthru_blockage_file file_name]] }
[ -name blockage_name [ -include_pin_patterns {pin_name_pattern_list} | -exclude_pin_patterns {pin_name_pattern_list} ]]
```

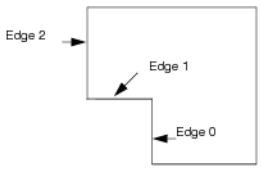
Creates a pin blockage for a module. You can run this command on the top-level module also. You can use this command during partition floorplanning.

Parameters

`-abutted_master_clone_channel_access_only`	Creates pin blockages on abutted edges of master-clones that abut to its parent or other master-clones. *Data_type*: bool, optional
`-area x1 y1 x2 y2`	Specifies the lower left x coordinate, lower left y coordinate, upper right x coordinate, and upper right y coordinate respectively of the blockage area. **Note:** The values of `x1`, `y1`, `x2`, and `y2` are relative to the top-level coordinates. *Data_type*: box, optional
`-cell cell_name`	Specifies the module for which the pin blockage will be created. This parameter is required if the module is not the top-level module. *Data_type*: string, optional
`-edge edge_number`	Specifies the integer value where edge numbering starts from lower-left corner of a partition clock-wise. Edge `0` is the edge that has the smallest y value. *Data_type*: int, optional
`-exclude_pin_patterns {pin_name_pattern_list}`	Specifies the pin-list on which the pin-blockage is not applicable. Pins which are specified in this list can be placed in this area legally. **Note**: The `-exclude_pin_patterns` and `-include_pin_patterns` options are mutually exclusive and cannot be used in a single command. Both these options require the use of the `-name` option to specify the name of the partition pin blockage. *Data_type*: string, optional
`-feedthru_blockage_file file_name`	Outputs a file with coordinates of blocked edges for pin assignment transformed into feedthru blockage_file format. *Data_type*: string, optional
`-help`	Outputs a brief description that includes the type and default information for each `create_pin_blockage` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man create_pin_blockage`
`-include_pin_patterns {pin_name_pattern_list}`	Specifies the pin-list on which the pin-blockage is applicable. All other pins can be placed in this area legally. **Note**: The `-exclude_pin_patterns` and `-include_pin_patterns` options are mutually exclusive and cannot be used in a single command. Both these options require the use of the `-name` option to specify the name of the partition pin blockage. *Data_type*: string, optional
`-layer {layer_id_list}`	Specifies the metal layer(s) for which pins are not to be created. **Note:** Layers can be specified using the LEF layer names or layer ID numbers. *Default:* If no layer number is specified, all metal layers are blocked. *Data_type*: string, optional

`-name` *`blockage_name`*	Specifies the name of the partition pin blockage. *Default:* By default, the name is `defPtnPinBlkName`. *Data_type*: string, optional
`-offset_end` *`distance`*	Specifies the distance, in microns, from the ending corner of the edge. *Minimum Value:* 0 *Maximum Value:* 65535 *Data_type*: float, optional
`-offset_start` *`distance`*	Specifies the distance, in microns, from the starting corner of the edge. *Minimum Value:* 0 *Maximum Value:* 65535 *Data_type*: float, optional

Examples

- The following command creates a pin blockage named `pinBlkg1` on the edge `0` of layers `Metal5` and `Metal7` for the module `PtnBlock1`.
  ```
  create_pin_blockage -edge 0 -name pinBlkg1 -cell PtnBlock1 -layer {Metal5 Metal7}
  ```
  

- The following command creates a pin blockage named `pinBlkg2` on layers Metal5 and Metal7 for the module `PtnBlock2`. The blockage area is bounded by the lower-left coordinates `100`, `200` and the upper-right coordinates `300`, `400`.
  ```
  create_pin_blockage -area 100 200 300 400 -name pinBlkg2 -cell PtnBlock2 -layer {Metal5 Metal7}
  ```
- The following command creates a pin blockage named pinBlkg3 that only applies to partition pins that match pattern1 and pattern2. All other partition pins can be placed in this area legally.
  ```
  create_pin_blockage -include_pin_patterns {*pattern1* *pattern2*} -cell PtnBlock3 -area $area -name pinBlkg3
  ```
- The following command creates a pin blockage named pinBlkg4 that only applies to all partition pins except those that match pattern1 and pattern2. Only pins that match pattern1 and pattern2 can be placed at this edge legally.
  ```
  create_pin_blockage -exclude_pin_patterns {*pattern1* *pattern2*} -cell PtnBlock4 -edge 0 -name pinBlkg4
  ```

Related Information

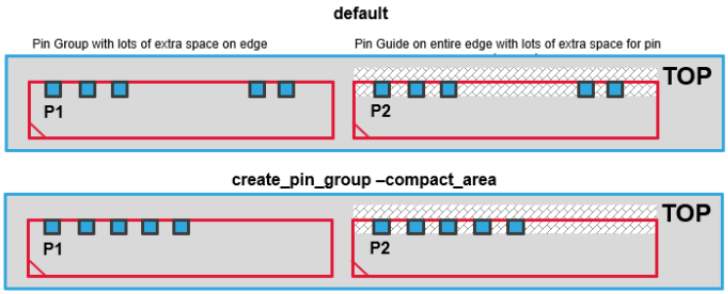
- `delete_pin_blockages`
- Partitioning the Design chapter in the *Innovus Stylus Common UI User Guide*
  - Setting Pin Constraints
  - Pin Blockage

create_pin_group

```
   create_pin_group
[-help]
[-exclude_pins {all_layer | same_layer | all_layer_in_guided_area}]
[-keep_out_spacing integer]
[-partition string]
[-spacing integer]
[-width width]
[ {[-opt_order ]} | -spread_pins | -compact_area ]
[ {-name pin_group_name [-pins {pin_name_list}]} ]
```

Creates a pin group for a partition or for the top-level. Pin groups in conjunction with pin guides are used to guide pin placement when the `assign_io_pins` or `assign_partition_pins` command is called. A pin group is a first class data object–delete with the `delete_pin_groups` command.

Parameters

`-compact_area`	Specifies that pins in the group are assigned tightly together. By default, pins which are part of a group and associated to a guide can be spread inside the guide based on the area available and alignment to targets. The `compact_area` option, when used, ignores the alignment and available area and forces the pins to the minimum area on guide honoring the minimum spacing constraint and overlooking any alignment to targets.  *Data_type*: bool, optional
`-exclude_pins {all_layer | same_layer | all_layer_in_guided_area }`	
	Specifies that pins in the group exclude foreign pins on all layers, same layer, or on all layers in the guided area. • `all_layer`: Specifies that pins in the group exclude foreign pins between grouped pins on all layers. • `same_layer`: Specifies that pins in the group exclude foreign pins between grouped pins on the same layer. • `all_layer_in_guided_area`: Specifies that pins in the group exclude foreign pins in the guide area defined for the group, on all layers. **Note**: If a guide is not defined for a pin group, `-exclude_pins all_layer_in_guided_area` option is treated like `-exclude_pins all_layer`. *Data_type*: enum, optional
`-help`	Outputs a brief description that includes type and default information for each `create_pin_group` parameter. For a detailed description of the command and all of its parameters, use the man command: `man create_pin_group`
`-keep_out_spacing integer`	Specifies the minimum spacing between the pins named in the pin group and any other pins, in number of tracks. *Minimum*: 0 *Maximum*: 65535 *Data_type*: int, optional
`-name pin_group_name`	Specifies the name of the pin group being created. *Data_type*: string, required

`-opt_order`	Specifies that pins in the group are to be reordered to optimize wire length during pin placement. Otherwise the pin order is exactly as specified in the pin group. *Data_type*: opt, optional
`-partition` *string*	Specifies the name of the module or submodule to which the pin group applies. This parameter is not required if the pin group being created is for the top level. *Data_type*: string, optional
`-pins` *port_name_list*	Specifies the pins to be added to the pin group. Additional pins can be added subsequently with the `update_pin_group` command. *Data_type*: string, optional
`-spacing` *integer*	Specifies the minimum spacing between the pins of the pin group, in number of tracks. *Default*: Two tracks. *Data_type*: int, optional
`-spread_pins`	Specifies that pins in the group are distributed evenly. *Data_type*: bool, optional
`-width` *width*	Specifies the pin width, in microns. *Data_type*: float, optional

Example

- The following command creates a pin group PG_A22 for module SH17 and adds the pins pinA22_1 and pinA22_2 to it.
  ```
  create_pin_group PG_A22 -partition SH17 -pins {pinA22_1 pinA22_2}
  ```
- The following command creates a pin group PG_A for module SH17 and adds the pins pinA22_1 and pinA22_2 to it. No other pins are to be placed between pinA22_1 and pinA22_2 in any layer.
  The pins may be reordered to optimize wire length.
  ```
  create_pin_group PG_A22 -partition SH17 -pins {pinA22_1 pinA22_2} -exclude_pins all_layer -opt_order
  ```
- The following command deletes the pin group named *myPinGroup*:
  ```
  delete_obj [ get_db pin_groups "myPinGroup" ]
  ```

Related Information

- `assign_io_pins`
- `assign_partition_pins`
- `delete_obj`
- `delete_pin_groups`
- `update_pin_group`
- `create_pin_guide`
- `assign_pins` Category Attributes
- Partitioning the Design chapter in the *Innovus Stylus Common UI User Guide*
  - Setting Pin Constraints
  - Pin Group

create_pin_guide

```
create_pin_guide
[-help]
{[ -area {x1 y1 x2 y2} | [-edge integer  [[-offset_start float] [-offset_end float]]] ] [ -layers {layeridList} [-layer_priority ]]}
 [-pin string | -pin_group string | -net string | -net_group string | -name string ]
[-edge integer | -net string | -net_group string]
[-partition string | -net string | -net_group string]
```

Creates a pin guide object. The pin guide restricts the pins associated with the pin guide within the specified area. You can run this command on the top-level module also. You can assign the same pin group or net group to multiple pin guides. Pin guides are generated for buses, nets, net groups, pins, and pin groups. For buses and nets that do not physically connect to the partition, pin guides will not be created. You can use this command during partition floorplanning.

Parameters

`-area x1 y1 x2 y2`	Specifies the coordinates of the pin guide area. • x1: Specifies the lower left x coordinate of the pin guide area. • y1: Specifies the lower left y coordinate of the pin guide area. • x2: Specifies the upper right x coordinate of the pin guide area. • y2: Specifies the upper right y coordinate of the pin guide area. *Data_type*: rect, required
`-edge integer`	Specifies the integer value where edge numbering starts from lower-left corner of a partition clockwise. Edge 0 is the edge that has the smallest y value. *Data_type*: int, optional
`-help`	Outputs a brief description that includes type and default information for each `create_pin_guide` parameter. For a detailed description of the command and all of its parameters, use the man command: `man create_pin_guide`.
`-layer_priority`	Assign pins based on the order of layers specified by the user. *Data_type*: bool, optional
`-layers {layerIdList}`	Specifies the metal layer(s) for the pin guide. *Default*: You must specify the correct pin layer. If you do not specify the pin layer, Innovus will display an error instead of choosing the preferred layer to assign pins like the old behavior. **Note**: Layers can be specified using the LEF layer names or layer ID numbers. *Data_type*: string, optional
`-name string`	Specifies the name of the partition pin guide object. If you specify this parameter, the command will automatically detect whether the object is a pin, pin group, net, net group, or bus. In case there are different objects with the same name, the command selects the object with the following precedence: • If a cell name is specified, the command selects the object with the following precedence: a. pin b. pin group c. net d. net group or bus • If a cell name is not specified, the command selects the object with the following precedence: a. net b. net group or bus For example, if the value of objectName is P1, and there is a pin group named P1 as well as a bus named P1, the command will select the pin group named P1. *Data_type*: string, optional

`-net string`	For nets, specifies the name of the net. For buses, specifies the name of the bus or the bus radix. *Data_type*: string, optional
`-net_group string`	Specifies the name of the net group. *Data_type*: string, optional
`-offset_end float`	Specifies the distance, in microns, from the ending corner of the edge. *Minimum Value*: 0 *Maximum Value*: 65535 *Data_type*: float, optional
`-offset_start float`	Specifies the distance, in microns, from the starting corner of the edge. *Minimum Value*: 0 *Maximum Value*: 65535 *Data_type*: float, optional
`-partition string`	Specifies the name of the cell. This parameter is not required if you are using the command for a top-level module. *Data_type*: string, optional
`-pin string`	Specifies the name of the pin. *Data_type*: string, optional
`-pin_group string`	Specifies the name of the pin group. *Default*: By default the name is `defRGuideName`. *Data_type*: string, optional

Example

- The following command creates a pin guide object for bus addr:
  ```
  create_pin_guide -area 2353.8000 7123.000 2653.8000 8298.2000 -name addr
  ```

Related Information

- `delete_pin_guides`
- Partitioning the Design chapter in the *Innovus Stylus Common UI User Guide*
  - Setting Pin Constraints
  - Pin Guides

create_pushdown_eco

```
  create_pushdown_eco
[-help]
[-generate_script_only]
[-eco_obj_dir outputDirName]
-original_dir originalDirName
[-partition_net_dir ptnNetDirName]
[-partition ptnNameList]
-pushdown_dir pushdownDirName
```

Creates logical and physical ECO files that are used to push down ECO objects by comparing two partition databases. It generates logical and physical ECO objects for each partition, in parallel, using EDP (parallel/distributed) framework. The regular/special wire ECO is generated for those nets which are specified via `-partition_net_dir`.

Note: A FlexHtree amendment file will also be created besides the logical ECO and DEF file.

Parameters

`-eco_obj_dir` *outputDirName*	Specifies the directory in which eco objects for each partition will be created. The eco objects are saved in sub-directories, one for each partition. *Default*: `./eco_obj_dir` *Data_type*: string, optional
`-generate_script_only`	Generates a TCL command file but does not execute it. *Data_type*: bool, optional
`-help`	Outputs a brief description that includes type and default information for each `create_pushdown_eco` parameter. For a detailed description of the command and all of its parameters, use the man command: `man create_pushdown_eco`.
`-original_dir` *originalDirName*	Specifies the directory in which original partition sub-directories are present. *Data_type*: string, optional
`-partition_net_dir` *ptnNetDirName*	Specifies the directory in which net file for each partition is present. The net file contains the nets for which regular/special wire ECO is generated. **Note**: Specifying `-partition_net_dir` iis not mandatory for the `create_pushdown_eco` command. If the `-partition_net_dir` parameter is not specified, then logically modified nets are selected for DEF ECO generation. Otherwise, a union of `-partition_net_dir` and logically-modified-nets is selected for DEF ECO generation. *Data_type*: string, optional
`-partition` *ptnNameList*	Specifies the list of partition names for which ECO objects are to be created. *Data_type*: string, optional
`-pushdown_dir` *pushdownDirName*	Specifies the directory in which pushdown (Example, clock tree constructed) version of partition sub-directories are present. These partitions will be compared with respective original partitions, and ECO objects will be generated for each of them. *Data_type*: string, required

Related Information

- `commit_pushdown_eco`
- `partition_push_network`
- Partitioning the Design chapter in the *Innovus Stylus Common UI User Guide*
  - Partition Pushdown Replay Flow

delete_all_partition_cuts

```
delete_all_partition_cuts
[-help]
```

Deletes all the partition cuts in the floorplan. You can use this command after specifying a partition.

Note: Command `delete_all_partition_cuts` is now obsolete and will be removed in the next major release of the software. This command has been replaced by the `delete_partition_cuts` command.

Parameters

`-help`	Outputs a brief description of the `delete_all_partition_cuts` command. For a detailed description of the command, use the `man` command: `man delete_all_partition_cuts`

Related Information

- `delete_partition_cuts`
- `delete_partitions`
- Partitioning the Design chapter in the *Innovus Stylus Common UI User Guide*

delete_all_partition_feedthrus

```
delete_all_partition_feedthrus
[-help]
```

Deletes all the partition feedthrough objects in the floorplan. You can use this command during partition floorplanning.

Parameters

`-help`	Outputs a brief description of the `delete_all_partition_feedthrus` command. For a detailed description of the command, use the `man` command: `man delete_all_partition_feedthrus`

Example

The following command removes all feedthrough objects from the floorplan:

```
delete_all_partition_feedthrus
```

Related Information

- `add_partition_feedthrus`
- `create_partition_feedthru`
- Partitioning the Design chapter in the *Innovus Stylus Common UI User Guide*
  - Inserting Feedthroughs

delete_black_boxes

```
delete_black_boxes
[-help]
{-cells cellName | -all}
[-keep_partition]
```

Deletes or unspecifies a blackbox and converts it to a module if the blackbox was originally defined as a module, or a block if the blackbox has a cdump or LEF macro definition. You can use this command after importing the design.

Note: You cannot delete a blackbox of an undefined module or cell.

Parameters

`-all`	Selects all blackboxes for deletion. *Data_type*: bool, optional
`-cells cellname`	Selects an individual blackbox for deletion. You can use wildcards (`*?`) with this parameter. *Data_type*: string, optional
`-help`	Outputs a brief description that includes the type and default information for each `delete_black_boxes` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man delete_black_boxes`
`-keep_partition`	Preserves the partition information. You should only use this option if the blackbox is placed inside the core area and is defined as a module in the netlist. *Data_type*: bool, optional

Example

The following command deletes all blackboxes that begin with the letter A – some of which are inside the core area, and some which are outside the core area and converts them to either a fence or module.

```
delete_black_boxes -cells A*
```

Related Information

- `create_blackbox`
- `convert_blackbox_to_fence`
- `convert_fence_to_black_box`
- `get_black_box_area`
- `read_black_box_netlist`
- `set_blackbox_master_orient`
- `set_blackbox_master_to_r0`
- Partitioning the Design chapter in the *Innovus Stylus Common UI User Guide*
  - Defining Blackboxes

delete_net_group

```
delete_net_group
[-help]
net_group_name
```

Removes a net group that was created earlier.

Note: When you delete a net group, any bus guide associated with the net group also gets deleted.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `delete_net_group` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man delete_net_group`
`net_group_name`	Specifies the name of the net group. You can use wildcards (`*?`) with this parameter. *Data_type*: string, required

Example

The following command removes netgroup `NG_22`.

```
delete_net_group NG_22
```

Related Information

- `create_net_group`
- `update_net_group`
- `delete_nets_from_net_group`
- Partitioning the Design chapter in the *Innovus Stylus Common UI User Guide*
  - Setting Pin Constraints
  - Net Group

delete_nets_from_net_group

```
delete_nets_from_net_group
[-help]
net_group
-delete_nets {netName | netNameList}
```

Removes a net from a net group. You can use this command after importing the design.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `delete_nets_from_net_group` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man delete_nets_from_net_group`
net_group	Specifies the name of the net group. You can use wildcards (`*?`) with this parameter. *Data_type*: string, required
`-delete_nets {`*netName* `|` *netNameList*`}`	Specifies the name of the net(s). You can use wildcards (`*?`) with this parameter. *Data_type*: string, required

Example

The following command removes net `NET1` from net group `NG22`:

```
delete_nets_from_net_group NG22 NET1
```

Related Information

- `create_net_group`
- `update_net_group`
- `delete_net_group`
- Partitioning the Design chapter in the *Innovus Stylus Common UI User Guide*
  - Setting Pin Constraints
  - Net Group

delete_partition_cuts

```
delete_partition_cuts
[-help]
{-partition {ptnName | ptnNameList} | -all}
```

Deletes all the partition cuts.

Parameters

`-all`	Deletes all the partition cuts. *Data_type*: bool, optional
`-help`	Outputs a brief description that includes type and default information for each `delete_partition_cuts` parameter. For a detailed description of the command and all of its parameters, use the man command: `man delete_partition_cuts`
`-partition {ptnName | ptnNameList}`	Specifies the name of the partition, or the list of partitions for which partition cuts are deleted. *Data_type*: string, optional

Example

The following command deletes partition cuts for partition `sheet17`:

```
delete_partition_cuts -partition sheet17
```

Related Information

- `delete_all_partition_cuts`
- `delete_partitions`
- Partitioning the Design chapter in the *Innovus Stylus Common UI User Guide*

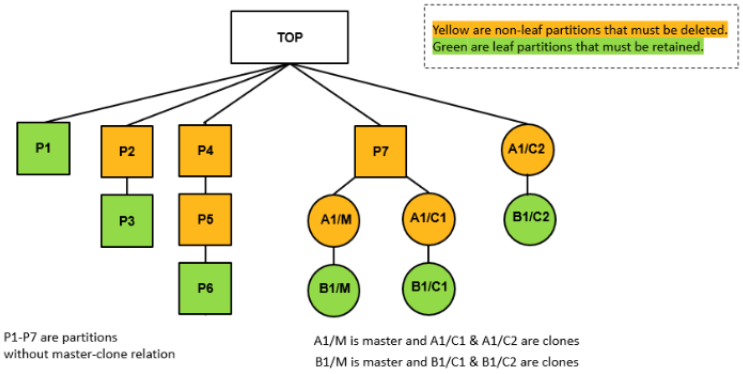
delete_partitions

```
delete_partitions
[-help]
{ partition_name | -all | -nested | -non_leaf }
[-undo_tcl_file file_name [-non_leaf ]]
```

Deletes the specified partition.

Note: The Hinst retains the fence property.

Parameters

`-all`	Deletes all partitions. *Data_type*: bool, optional
`-help`	Outputs a brief description that includes the type and default information for each `delete_partitions` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man delete_partitions.`
`-nested`	Deletes all nested partitions in the design. *Data_type*: bool, optional
`-non_leaf`	Deletes all non-leaf partitions in the design. **Note:** It saves the information of the deleted partitions in a tcl file, so that the same can be used to restore the deleted partitions. For example, in the diagram below, all the non-leaf partitions (marked in yellow) must be deleted and dumped in a tcl file for restoration p  *Data_type*: bool, optional
partition_name	Specifies the name of the partition to delete. *Data_type*: string, optional
`-undo_tcl_file` *fileName*	
	Specifies the file that can be sourced to undo deletion of partitions. With this parameter, the partition definition of the partition being deleted is written to the specified file. Later, if required, this file can be so the partition definition. *Data_type*: string, optional

Example

- The following command deletes the partition `p1`.
  ```
  delete_partitions p1
  ```
- The following command writes the partition definition of partition p1 in file ptn.tcl and then deletes it.
  ```
  delete_partitions p1 -undo_tcl_file ptn.tcl
  ```

Related Information

- `delete_partition_cuts`
- `delete_all_partition_cuts`
- `commit_partitions`
- Partitioning the Design chapter in the *Innovus Stylus Common UI User Guide*

delete_pin_blockages

```
delete_pin_blockages
[-help]
{-name pinBlkgName | -all}
```

Deletes a pin blockage or all pin blockages.

Parameters

`-all`	Specifies that all pin blockages will be deleted. *Data_type*: bool, optional
`-help`	Outputs a brief description that includes the type and default information for each `delete_pin_blockages` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man delete_pin_blockages`
`-name pinBlkgName`	Specifies the name of the pin blockage. You can specify wildcards (?*) with this parameter. *Data_type*: string, optional

Example

The following command deletes the pin blockage `PtnPinBlkg1`:

```
delete_pin_blockages -name PtnPinBlkg1
```

Related Information

- `create_pin_blockage`
- Partitioning the Design chapter in the *Innovus Stylus Common UI User Guide*
  - Setting Pin Constraints
  - Pin Blockage

delete_pin_from_pin_group

```
delete_pin_from_pin_group
[-help]
[-partition cellName]
-pin_group pinGroupName
-pins {pinName | pinNameList}
```

Removes a port from a specified pin group.

Parameters

`-partition` *`cellName`*	Specifies the name of the module. The module name is not required if the command is run on a top-level module. *Data_type*: string, optional
`-help`	Outputs a brief description that includes the type and default information for each `delete_pin_from_pin_group` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man delete_pin_from_pin_group`
`-pin_group` *`pinGroupName`*	Specifies the name of the pin group that was created. *Data_type*: string, required
`-pins {`*`pinName`* `|` *`pinNameList`*`}`	Specifies the name(s) of the port(s) of the module. *Data_type*: string, required

Example

The following command removes ports `RASN1` and `RASN2` of module `SH17` from pin group `pin17`:

```
delete_pin_from_pin_group -partition SH17 -pin_group pin17 -pins {RASN1 RASN2}
```

Related Information

- `update_pin_group`
- `create_pin_group`
- `delete_pin_groups`
- Partitioning the Design chapter in the *Innovus Stylus Common UI User Guide*
  - Setting Pin Constraints
  - Pin Group

delete_pin_groups

```
delete_pin_groups
[-help]
[-cell cellName]
-pin_group pinGroupName
```

Deletes a pin group or all pin groups that were created earlier.

Parameters

`-cell cellName`	Specifies the name of the module. The module name is not required if the command is run on a top-level module. *ata_type*: string, optional
`-help`	Outputs a brief description that includes the type and default information for each `delete_pin_groups` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man delete_pin_groups`
`-pin_group pinGroupName`	Specifies the name of the pin group that was created. *Data_type*: string, required

Example

The following command deletes the pin group `PinGroup17` of module `SH17`:

```
delete_pin_groups -cell SH17 -pin_group PinGroup17
```

Related Information

- `update_pin_group`
- `create_pin_group`
- `delete_pin_from_pin_group`
- Partitioning the Design chapter in the *Innovus Stylus Common UI User Guide*
  - Setting Pin Constraints
  - Pin Group

delete_pin_guides

```
delete_pin_guides
[-help]
{ -pin pin_name | -pin_group pin_group_name | -net net_name | -net_group net_group_name | -name object_name | -all }
[-partition ptn_name | -net net_name | -net_group net_group_name]
```

Deletes a pin guide object in the floorplan. You can use this command during partition floorplanning.

Parameters

`-all`	Specifies that all pin guides will be deleted. *Data_type*: bool, optional
`-partition` *ptn_name*	Specifies the name of the module. The module name is not required if the command is run on a top-level module. *Data_type*: string, optional
`-help`	Outputs a brief description that includes the type and default information for each `delete_pin_guides` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man delete_pin_guides`
`-name` *object_name*	Specifies the name of the partition pin guide object. If you specify this parameter, the command will automatically detect whether the object is a pin, pin group, net, net group, or bus. In case there are different objects with the same name, the command selects the object with the following precedence: • If a cell name is specified, the command selects the object with the following precedence: 1. pin 2. pin group 3. net 4. net group or bus • If a cell name is not specified, the command selects the object with the following precedence: 1. net 2. net group or bus For example, if the value of *objectName* is `P1`, and there is a pin group named `P1` as well as a bus named `P1`, the command will select the pin group named `P1`. *Data_type*: string, optional
`-net` *net_name*	For nets, specifies the name of the net. For buses, specifies the name of the bus or the bus radix. *Data_type*: string, optional
`-net_group` *net_group_name*	Specifies the name of the net group. *Data_type*: string, optional
`-pin` *pin_name*	Specifies the name of the pin. *Data_type*: string, optional
`-pin_group` *pin_group_name*	Specifies the name of the pin group. *Default:* By default the name is `defRGuideName`. *Data_type*: string, optional

Example

The following command deletes a pin guide object named `group_1`:

```
delete_pin_guides -name group_1
```

Related Information

- `create_pin_guide`
- Partitioning the Design chapter in the *Innovus Stylus Common UI User Guide*

edit_pin

```
edit_pin
[-help]
[-end {x y}]
[-fixed_pin]
[-global_location]
[-include_rectilinear_edge]
[-master_clone_aware]
[-offset_end value]
[-offset_start value]
-pin {pinName | pinNameList}
[-pin_depth pinDepthValue]
[-pin_width pinWidthValue]
[-quiet]
[-skip_wrapping_pins]
[-snap {track | usergrid | mgrid}]
[-spread_direction {clockwise | counterclockwise | both}]
[-start { x y}]
[-use { signal | clock | analog }]
[ {-layer {layerId | layerIdList} | {[-layer_horizontal layerId] [-layer_vertical layerId]} }
[-layer_priority ]
[-assign {x y} |
-spread_type {start | center | side | edge | range} |
-pattern {fill_track | fill_layer | fill_optimised | fill_diagonal | fill_sinusoidal |fill_checkerboard}]]
[-side sideName | -edge edgeNumber]
[-cell cellName | {-hinst hinstName  [-ref_master]} ]
[-fix_overlap [ -honor_constraint ]]
[-pattern {fill_track | fill_layer | fill_optimised | fill_diagonal | fill_sinusoidal | fill_checkerboard}
[-reverse_alternate ]]
[-spacing spacingValue [ -unit {micron | track} ]]
```

Modifies properties of pins, such as pin spreading, pin location, pin width and depth, spacing, snap-to location, and status. The `edit_pin` command provides the equivalent capabilities of the Pin Editor form.

Note: The `edit_pin` command does not support wildcards.

The `edit_pin` command honors the minimum area rule as follows:

- If neither the `-pin_width` nor the `-pin_depth` parameters have been specified, the `edit_pin` command takes the minimum pin width and the minimum area from the LEF file and calculates the pin depth accordingly.
- If only the `-pin_width` parameter has been specified, the `edit_pin` command takes the minimum area from the LEF file and calculates the pin depth accordingly.
- If only the `-pin_depth` parameter has been specified, the `edit_pin` command takes the minimum area from the LEF file and calculates the pin width accordingly.

If both the `-pin_width` and the `-pin_depth` parameters are specified, the `edit_pin` command takes these specified values, without enforcing the minimum area rule.

The `edit_pin` command is clone context aware. The command:

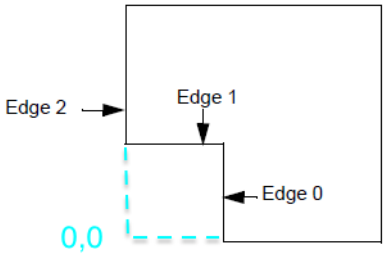
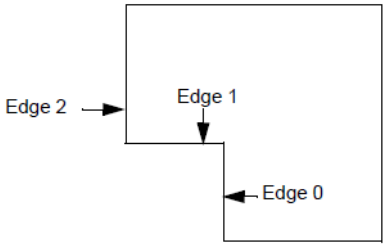
- Looks at routing blockages on clone and honors them while keeping pins of master with `-cell <>`.
- Looks at abutted macro blockages and honors them while keeping pins of master with `-cell <>`.
- Honors route blockage on master.

Note: The value of the parameters of the `edit_pin` command is not stored across multiple runs of the command – you need to specify the required value with every run of the command.
For example, if you do not specify the `-pin_width` parameter, the default pin width of the specified layer is used.

Note: The `edit_pin` command checks and corrects DRC violations and overlapping while placing internal pins. It supports pin placement on Metal 1 and accepts all layers for pin placement by default.

Note: The `edit_pin` command is used for manual pin assignment in nested partitions. It places pins of nested partitions similar to single level partitions.

Parameters

`-assign {x y}`	Pre-assigns a pin at the specified location. You can use this parameter with other parameters such as `-snap`, `-fixOverla` **Note**: `-assign {x y}` considers the local coordinates from the bounding box of the fence.  *Data_type*: point, optional
`-cell cellName`	Specifies the name of the module. If you do not specify a cell name, the command runs on the top-level cell. *Data_type*: string, optional
`-edge edgeNumber`	For rectilinear partitions, specifies the edge along which the pins will be placed. The edge number is an integer. The edg Edge 0 is the edge that has the smallest y value.  To spread pins on multiple rectilinear edges that are on the same side, use the `-side` parameter. For rectangular partitions, you can specify a side with the -side parameter. *Data_type*: int, optional
`-end {x y}`	For pin spreading along a range, specifies the coordinates where pins should end. This parameter is used with the `-spr` *Data_type*: point, optional
`-fixed_pin`	A value of 1 sets the pin status to Fixed. *Default*: The pin status is placed. *Data_type*: bool, optional
`-fix_overlap {0 | 1}`	With this parameter on, the `edit_pin` command tries to resolve pin-overlap with other/nearby objects. The pins are moved to the closest acceptable position such that pins do not overlap. *Default*: 1. Resolve pin-overlap is on. **Note**: This parameter applies only when pins are snapped to layer track. *Data_type*: bool, optional

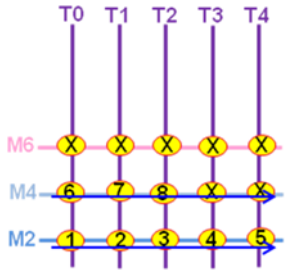
`-global_location`	Specifies that the global location should be considered by the command.
	Note: By default all locations that the `edit_pin` command considers are coordinates with respect to lower-left corner of pa accept coordinates with respect to the lower-left corner of the design.
	For example, in the following command, the pin will be placed 3u from the lower-left corner of the partition.
	`edit_pin -cell c -pin cIn -assign 0 3 -layer 2 -side left`
	When you use the `-global_location` parameter, the pin will be placed 3u from the lower-left corner of the design.
	`edit_pin -cell c -pin cIn -assign 0 3 -layer 2 -side left -global_location`
	Data_type: bool, optional

`-help`	Outputs a brief description that includes type and default information for each `edit_pin` parameter. For a detailed description of the command and all of its parameters, use the man command: `man edit_pin`
`-hinst hinstName`	Specifies the hinst name, including black box or committed partition and top. *Data_type*: string, optional
`-honor_constraint {0|1}`	Specifies whether user-defined pin constraints should be honored. The following constraints are honored when this parameter is set to on: • pin spacing • pin size • pin-to-corner distance • partition-allowed pin layers. *Default*: 0 (user-defined pin constraints are not honored) *Data_type*: bool, optional
`-include_rectilinear_edge`	Specifies that all the edges coming in the solution space should be included. *Data_type*: bool, optional
`-layer {layerId | layerName}`	Specifies the layer on which the pins will be assigned. Layers can be specified using the LEF layer names or layer ID nu **Note**: This parameter is optional but only when `-spread_type` is not specified. **Note:** For designs that have backside metal layers, both the backside and frontside layers are accepted. *Data_type*: bool, optional
`-layer_horizontal layerId`	Specifies the preferred horizontal layer on which the pins will be assigned. **Note:** For designs that have backside metal layers, both the backside and frontside layers are accepted. *Data_type*: string, optional
`-layer_vertical layerId`	Specifies the preferred vertical layer on which the pins will be assigned. **Note:** For designs that have backside metal layers, both the backside and frontside layers are accepted. *Data_type*: string, optional
`-layer_priority`	Specifies that the input layer is given based on its priority. *Data_type*: bool, optional
`-master_clone_aware`	Enables master-clone context awareness for feasibility checking. *Data_type*: bool, optional
`-offset_end value`	Sets the offset from the end of the edge (in microns) to be considered as the endpoint for placing the pin. **Note**: The `-offset_start` and `-offset_end` options can only be used with the `-edge` parameter. *Data_type*: float, optional
`-offset_start value`	Sets the offset from the start of the edge (in microns) to be considered as the start point for placing the pin. **Note**: The `-offset_start` and `-offset_end` options can only be used with the `-edge` parameter. *Data_type*: float, optional

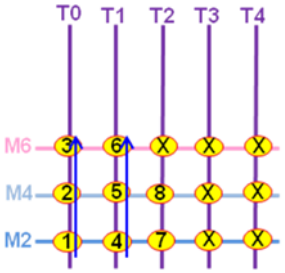
`-pattern {fill_track | fill_layer | fill_optimised | fill_diagonal | fill_sinusoidal | fill_checkerboard}`

Specifies the multi-layer-spread-pattern must be followed by the set of selected pins.

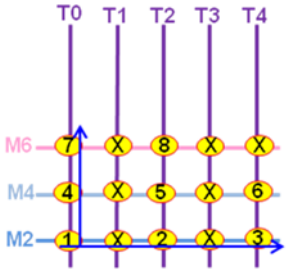
fill_track



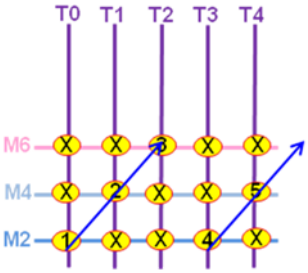
fill_layer



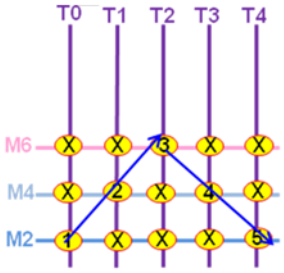
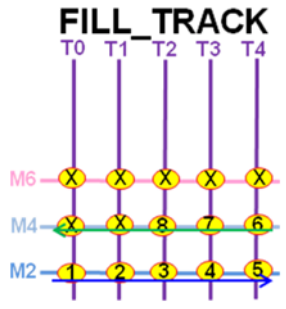
fill_optimised



fill_diagonal



fill_sinusoidal

	 fill_checkerboard
`-pin {pinName | pinNameList}`	Specifies the pins, or the list of pins of the specified module on which the command will run. **Note**: The naming convention to identify individual ports is based on the order in which the ports are stored inside the da port2, respectively. For example: VDD.port0, VDD.port1, VDD.port2 *Data_type*: string, required
`-pin_depth pinDepthValue`	Sets the depth of the pin in microns. *Default*: If you do not specify this parameter, the default depth of the specified layer is used. *Data_type*: float, optional
`-pin_width pinWidthValue`	Sets the width of the pin in microns. *Default*: If you do not specify this parameter, the default width of the specified layer is used. *Data_type*: float, optional
`-quiet`	Specifies that the messages should be suppressed. *Data_type*: bool, optional
`-ref_master`	Specifies that the clone's edge, spreadDirection, and offset are with respect to the master. *Data_type*: bool, optional
`-reverse_alternate`	Specifies that the reverse of the multi-layer-spread-pattern must be followed by the set of selected pins. For example, `fill_track -reverse_alternate`  *Data_type*: bool, optional

`-side sideName`	For rectangular partitions, specifies the side on which the pins will be placed. The values for `sideName` can be: top, botton **Note**: The value inside can only be specified with the `-assign` parameter. For rectilinear partitions, you can also specify an edge with the `-edge` parameter. **Note:** The `set_db assign_pins_edit_in_batch` attribute does not support the internal pins (where the side is INSIDE). *Data_type*: string, optional
`-skip_wrapping_pins`	Skips pin wrapping on the adjacent edge or the same edge from the other side. *Data_type*: bool, optional
`-snap {track | usergrid | mgrid}`	
	Specifies how the pins will be snapped. • `track`: Snaps pins to layer tracks. • `usergrid`: Snaps pins to the user-defined grid. • `mgrid`: Snaps pins to the manufacturing grid. *Default*: `track` *Data_type*: enum, optional
`-spacing spacingValue`	Displays the edge-track-to-edge-track spacing between pins for multi-pin editing. The spacing is either in micrometers o placed on every track with no empty tracks between them. Default: The spacing is set automatically. *Data_type*: string, optional
`-spread_direction {clockwise | counterclockwise | both}`	
	Specifies the direction in which the pins are spread. You can specify: • `clockwise`: Sets the pins in the clockwise direction. • `counterclockwise`: Sets the pins in the counter clockwise direction. • `both`: Sets the pins by looking for the closest legal location to the desired location in both directions. *Default*: `clockwise` *Data_type*: enum, optional
`-spread_type {start | center | side | edge | range}`	
	Specifies how to spread pins along a block's edge. • `start`: Uses the coordinates of the first pin specified with the `-pin` parameter as the starting point (anchor) for spreadi parameter. • `center`: If a single pin is specified with the `-pin` parameter, the pin will be placed at the center of the specified edge/sic points such that the center pin(s) of the group are placed at the center of the specified edge or side. • `side`: If you specify a side with the `-side` parameter and specify the `-spread_type side` parameter, the pins are spread and from bottom to top. • `edge`: If you specify an edge with the `-edge` parameter and specify the `-spread_type edge` parameter, the pins are spre points. • `range`: Spreads pins evenly along the block edge between points you specify with `-start {x y}` and `-end {x y}` para Editor determines the appropriate spacing. For rectilinear partitions, if the start and/or end points are not on the specified side, the start and/or end points are snappe *Data_type*: enum, optional
`-start {x y}`	For pin spreading along a range, specifies the coordinates where pins should start. This parameter is used with the `-spr` *Data_type*: point, optional

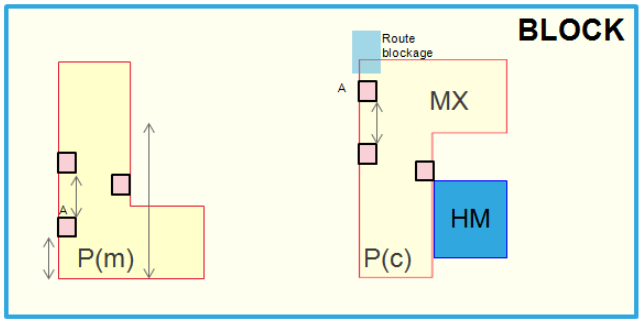
`-unit {micron | track}`	Specifies the unit in microns or in tracks. This parameter applies only to the unit of the -spacing parameter. *Default*: `micron` *Data_type*: enum, optional
`-use {signal | clock | analog}`	
	Specifies the +USE property of the partition pin. *Data_type*: stringl, optional

Example

The following command modifies the pins as specified here:

- the command runs on the module DTMF_CORE_INST1
- the command runs on all pins whose name starts with DTMF_CORE_INST1_OUTPIN
- the pins are spread on the edge number 2
- the use property is set to clock
- the pins are spread between the coordinates (17 23) and (117 123)
- the pins are assigned on the layer Metal3
- the pin status is set to Fixed
- the pins are snapped to layer tracks
- the unit is in tracks
- the spacing between pins for multi-editing is set to 2 tracks
- the pin depth is set to 10 microns.

  Note: The `-unit` parameter is set to track but that does not affect the unit of pin depth.
- the pins are moved to the nearest acceptable location as the `-fix_overlap` parameter is set to 1. However, the `-overlap_constraints` parameter is not specified.
  edit_pin
  ```
  -cell DTMF_CORE_INST1
  -pin {DTMF_CORE_INST1_OUTPIN}
  -spread_type range
  -edge 2
  -layer Metal3
  -use clock
  -fix_overlap 1
  -snap track
  -pin_depth 10
  -start {17 23}
  -end {117 23}
  -spacing 2
  -unit track
  -fixed_pin 1
  ```
- The `edit_pin` command looks at routing blockages on clone and honors them. In the figure below, the following command would not start pins on master from Y=0 as there is a route blockage on clone at desired location.
  ```
  edit_pin -hinst p(m) -spread_type start -start 0 0 -edge 0 -layer 2 -pin A*
  ```
  Also, the following command would not start pins on master from Y=10 as there is a macro with obstruction on clone at desired location.
  ```
  edit_pin -hinst p(m) -assign 10 5 -edge 2 -layer 2 -pin B -spread_direction {both}
  ```



- The following commands are used to edit pins in the batch mode:

```
set_db assign_pins_edit_in_batch true
edit_pin -hinst p(m) -spread_type start -start 0 0 -edge 0 -layer 2 -pin A*
...
...
edit_pin -hinst p(m) -assign 10 5 -edge 2 -layer 2 -pin B -spread_direction {both}
set_db assign_pins_edit_in_batch false
```

Related Information

- `assign_partition_pins`
- `assign_io_pins`
- `assign_pins` Category Attributes
- Floorplanning the Design chapter in the *Innovus Stylus Common UI User Guide*
  - Using the Pin Editor
- Partitioning the Design chapter in the *Innovus Stylus Common UI User Guide*
  - Assigning Pins
- Edit Menu chapter in the *Innovus Stylus Common UI Menu Reference*
  - Pin Editor

fix_feedthru_assign

```
fix_feedthru_assign
[-help]
-buffer string
[-delete_feed]
```

Fixes the assign statements that are associated with partitions by replacing them with the specified buffer. This helps to minimize the number of buffers that are added to the design when assign statements are deleted.

The `fix_feedthru_assign` command enables selective replacement of assign statements with real buffers.

Parameters

`-buffer` *`string`*	Specifies the buffer to be used for replacing assign statements. *Data_type*: string, required
`-delete_feed`	Removes input-output(s) assigns from within the partition module. *Data_type*: bool, optional
`-help`	Outputs a brief description that includes the type and default information for each `fix_feedthru_assign` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man fix_feedthru_assign`

Example

The following command replaces the assign statements associated with partitions with the specified buffer.

```
fix_feedthru_assign -buffer abc
```

Related Information

- `delete_assigns`
- `add_partition_feedthrus`
- `create_partition_feedthru`
- Partitioning the Design chapter in the *Innovus Stylus Common UI User Guide*

flatten_cover_cell

```
flatten_cover_cell
[-help]
[-inst {instanceName | instanceNameList}]
```

Dissolves and flattens any instance whose master has class `COVER`, but no subclass `BUMP`. You can use this command after loading a DEF file (`read_def`).

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `flatten_cover_cell` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man flatten_cover_cell`
`-inst {instanceName | instanceNameList}`	
	Specifies the instance, or a list of instances, to be flattened. If you are specifying a list, separate the instance names with a space. *Default*: If you do not specify this parameter, all instances whose masters have class `COVER` are flattened. *Data_type*: string, optional

Examples

- The following command flattens the cover cells named `coverCell1` and `coverCell2`
  ```
  flatten_cover_cell -inst {coverCell1 coverCell2}
  ```
- The following command flattens *all* cover cells
  ```
  flatten_cover_cell
  ```

Related Information

- `read_def`
- Partitioning the Design chapter in the *Innovus Stylus Common UI User Guide*

flatten_partitions

```
flatten_partitions
[-help]
[partition_name]
[-bring_back_row]
[-keep_pin_geometry]
[-no_inherit_physical]
```

Flattens (uncommits) the partition back to normal module status. This command should also be used to unassign blackbox pins *after* the blackbox is committed. To unassign blackbox pins *before* the blackbox is committed, use the `set_partition_pin_status` command. This command unassigns partition pins as well.

The `flatten_partitions` command supports partitions with flexmodels. It restores back flexmodel information if the committed partition has a flexmodel netlist.

To bring back row information for a partition, use the `-bring_back_row` parameter. If the partition rows are non-integer-multiple-height rows or are not aligned with the top-level rows, the `flatten_partitions` command will automatically create a power domain associated with the partition. The power domain will have the following characteristics:

- The timing library will the same as that for the top level.
- If a global net connection is defined in the partition block, the power domain will have the same global net connection. Otherwise the global net connection for the power domain will be set to the top-level net connection.
- The minimum gap values will be the same as the placement halo values for the partition.

The recommended use model for bringing back non-integer-multiple height rows or unaligned rows is as follows.

Top-down flow:

1. Create a power domain at full chip-level design.
2. Specify the same hierarchical instance as the partition

Bottom-up flow:

1. Create a power domain at top-level design.
2. Assign the instance block as a member of the power domain created in step 1.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `flatten_partitions` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man flatten_partitions`
`-bring_back_row`	Specifies that partition row information should be brought back for cases where: • the partition is not a power domain and row information in partition is different from top-level design, *and* • after flattening the partition, the rows of the partition space are not aligned with the top-level rows When you use this parameter, a power domain is created for the partition. *Default:* If you do not specify this parameter, the row information is *not* brought back in such cases. *Data_type*: bool, optional
`-keep_pin_geometry`	Specifies that for every partition pin, a DRCFILL metal of the type special route is placed on top of the partition pins. *Data_type*: bool, optional
`-no_inherit_physical`	Specifies that physical only cell instances are removed when partitions are flattened. *Data_type*: bool, optional
`partition_name`	Specifies the name(s) of the partition(s). You can specify any number of partitions. *Data_type*: string, optional

Example

The following command uncommits the partition status of partitions `sheet17` and `sheet25`:

```
flatten_partitions sheet17 sheet25
```

Related Information

- `assemble_design`
- `set_partition_pin_status`
- `commit_partitions`
- Partitioning the Design chapter in the *Innovus Stylus Common UI User Guide*
- Floorplanning the Design chapter in the *Innovus Stylus Common UI User Guide*

get_black_box_area

```
  get_black_box_area
[-help]
[-cell_density]
-inst cellName
[-macro_area]
[-stdcell_area]
```

Retrieves the standard cell area, macro area, and cell utilization value for the specified blackbox. If only the blackbox cell name is specified, the command reports the total blackbox area.

Note: You can use this command to report the black box area only if you had specified the blackbox using the `create_blackbox` command with the `-gate_area` parameter.

Parameters

`-cell_density`	Reports the standard cell utilization value for the blackbox. The standard cell utilization is calculated as follows: `standard cell area / (blackbox area - total macro area)` This parameter cannot be used if the blackbox was specified based on the total area or the gate count. *Data_type*: bool, optional
`-help`	Outputs a brief description that includes the type and default information for each `get_black_box_area` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man get_black_box_area`
`-inst cellName`	Specifies the name of the blackbox cell. *Data_type*: string, required
`-macro_area`	Reports the area of the macros included in the blackbox area. This parameter cannot be used if the blackbox was specified based on the total area or the gate count. *Data_type*: bool, optional
`-stdcell_area`	Reports the area of the standard cell included in the blackbox area. This parameter cannot be used if the blackbox was specified based on the total area or the gate count. *Data_type*: bool, optional

Examples

- The following command retrieves the total blackbox area for the blackbox `bbox_inst1`:
  ```
  get_black_box_area -inst bbox_inst1
  ```
- The following command retrieves the standard cell area, macro area, and cell utilization value for the blackbox `bbox_inst2`:
  ```
  get_black_box_area -inst bbox_inst2 -stdcell_area -macro_area -cell_density
  ```

Related Information

- `create_blackbox`
- `delete_black_boxes`
- Partitioning the Design chapter in the *Innovus Stylus Common UI User Guide*
  - Defining Blackboxes

get_clone_partition_orient

```
get_clone_partition_orient
[-help]
hinst
```

Retrieves the orientation information of the specified partition clone. You can use this command after using the `set_clone_partition_orient` command.

Parameters

`-help`	Outputs a brief description of the `get_clone_partition_orient` command. For a detailed description of the command, use the `man` command: `man get_clone_partition_orient`
`hinst`	Specifies the hinst of the partition clone whose orientation information has to be retrieved. This can be an hierarchical instance hinst of a partition or an instance hinst of a black box. **Note:** The hinst argument should be registered as a hinst and not as a string to support DPO usage. *Data_type*: (hinst), required

Example

The following command retrieves the the orientation information for the partition clone `TDSP_ARB_C1`:

```
get_clone_partition_orient TDSP_ARB_C1
```

Related Information

- `set_clone_partition_orient`
- Partitioning the Design chapter in the *Innovus Stylus Common UI User Guide*
  - Changing Partition Clone Orientation

get_partition_pin_status

```
get_partition_pin_status
[-help]
[-partition cellName]
[-pins {pinName | pinNameList}]
```

Retrieves the pin status that was defined with the `set_partition_pin_status` command.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `get_partition_pin_status` parameter. For a detailed description of the command and all of its parameters, use the man command: `man get_partition_pin_status`.
`-partition cellName`	Specifies the name of the partition. You can use wildcards (`*?`) with this parameter. *Data_type*: string, optional
`-pins pinName | pinNameList`	
	Specifies the name of the pin(s) of the partition. You can use wildcards (`*?`) with this parameter. *Default:* If the pin name is not specified, the command is run on all pins. *Data_type*: string, optional

Example

The following command tells the status of pin `DATA` of partition `EUG`:

```
get_partition_pin_status -pins DATA -partition EUG
```

Related Information

- `set_partition_pin_status`
- Partitioning the Design chapter in the *Innovus Stylus Common UI User Guide*

get_pin_constraint

```
get_pin_constraint
[-help]
[-dict_file string]
[[-global | -cell cell_name]
[-global | -pins pin_name_list]]
```

Retrieves information about the constraints set for a partition pin or an I/O pin using the `set_pin_constraint` command. It displays the information in a tabular format output report that includes constraints from other sources (top/bottom preferred layer and NDR-rule from `set_route_attributes`, reserved layer for partition, pin/net-group /guide constraints presence)) and segregates between user-defined, default, and lef values.

Parameters

`-cell cell_name`	Specifies the name of the module. The module name is not required if the command is run on a top-level module. You can use wildcards (*?) with this parameter. *Data_type*: string, optional
`-dict_file string`	Writes the output of the command in the specified text file (in dictionary format). *Data_type*: string, optional
`-global`	Specifies to get global constraints. *Data_type*: bool, optional
`-help`	Outputs a brief description that includes the type and default information for each `get_pin_constraint` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man get_pin_constraint`
`-pins pin_name_list`	Specifies the name(s) of the partition pin(s). You can use wildcards (`*?`) with this parameter. *Data_type*: string, optional

Example

- The following command prints all the constraints set for the specified module.

```
get_pin_constraint -cell dtmf_chip
```

  The following result is displayed on the screen:

```
Default Global Values:
----------------------
Minimum pin-to-pin track spacing : 2
Pin to corner distance : 5

Template from tech LEF
------------------------------
| Layer | width | depth |
------------------------------
| 2 | 0.2800 | 0.2800 |
| 3 | 0.2800 | 0.2800 |
| 4 | 0.2800 | 0.2800 |
| 5 | 0.2800 | 0.2800 |
| 6 | 0.4400 | 0.4400 |
------------------------------

Global Constraint:
-------------------

Constraint on partition dtmf_chip :
-----------------------
Partition reserved layers { 1 2 3 4 5 6 7 }
--------------------------
| Side | allowed layer |
--------------------------
| top | 2 4 6 |
| left | 3 5 |
| bottom | 2 4 6 |
| right | 3 5 |
--------------------------
```

Related Information

- set_pin_constraint
- reset_pin_constraint
- Partitioning the Design chapter in the *Innovus Stylus Common UI User Guide*
  - Setting Pin Constraints

gui_highlight_feedthru_nets

```
gui_highlight_feedthru_nets
[-help]
net_map_file
[ net ]
```

Highlights all nets for which feedthrough buffers were inserted with the `add_partition_feedthrus` command. The highlighted feedthrough path consists of the nets, the terms that the nets connect to, and the instances that contain those terms.

Note: For the `gui_highlight_feedthru_nets` command to work, the `add_partition_feedthrus` command must be run with the `-net_mapping` parameter. The net mapping file generated with the `add_partition_feedthrus -net_mapping` parameter is used by the `gui_highlight_feedthru_nets` command to highlight the feedthrough nets.

To dehighlight the feedthrough nets, run the `gui_clear_highlight` command.

Note: It displays a warning message when used for a design with nested partitions and generates results based on single level partitions.

Parameters

net_map_file	Specifies the net mapping file that was generated with the `add_partition_feedthrus -net_mapping` parameter. *Data_type*: string, required
-help	Outputs a brief description that includes the type and default information for each `gui_highlight_feedthru_nets` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man gui_highlight_feedthru_nets`
net	Specifies the name of the net(s) whose feedthrough path should be highlighted. *Default*: The feedthrough paths of all nets specified in the net mapping file are highlighted. *Data_type*: string, optional

Examples

- The following command highlights all feedthrough nets specified in the net mapping file `nmpFile1`.
  ```
  gui_highlight_feedthru_nets nmpFile1
  ```
- The following command highlights all feedthrough nets that are specified in the net mapping file nmpFile1 whose name begins with `feedthruNet_core`.
  ```
  gui_highlight_feedthru_nets nmpFile1 feedthruNet_core*
  ```

Related Information

- `add_repeater_by_rule`
- `add_partition_feedthrus`
- `gui_show_partition_wire_cross_over`
- `create_partition_feedthru`
- `delete_all_partition_feedthrus`
- Partitioning the Design chapter in the *Innovus Stylus Common UI User Guide*
  - Inserting Feedthroughs
  - Highlighting the Nets for which Feedthrough Buffers Have been Inserted

gui_show_partition_wire_cross_over

```
gui_show_partition_wire_cross_over
[-help]
[partition_name]
[-out_file fileName]
[-exclude_net exNetFileName]
[-exclude_clock]
[-exclude_tristate]
[-exclude_high_fanout nrFan]
[-all_nets]
[-delta]
```

Displays and writes a file of wires that physically cross over partitions. You can use this command after routing the design.

Note: The `gui_show_partition_wire_cross_over` command supports designs with nested partitions.

Parameters

`-all_nets`	Reports nets that cross partitions that also connect to the ports of partitions. If this parameter is not specified, only nets that cross the partitions are reported. *Data_type*: bool, optional
`-delta`	Reports the total segment length of the wires crossing the block in the X and Y directions. *Data_type*: bool, optional
`-exclude_clock`	Excludes clock nets from the wire crossing information. *Data_type*: bool, optional
`-exclude_high_fanout` *nrFan*	Excludes nets with fanouts over nrFan from the display. *Data_type*: int, optional
`-exclude_net` *exNetFileName*	Specifies the filename that contains nets to be excluded from the display. *Data_type*: string, optional
`-exclude_tristate`	Excludes all tristate nets from the display. *Data_type*: bool, optional
`-help`	Outputs a brief description that includes the type and default information for each `gui_show_partition_wire_cross_over` parameter. For a detailed description of the command and all of its parameters, use the man command: `man gui_show_partition_wire_cross_over`
`-out_file` *fileName*	Specifies the name of the output file of wire crossing information. *Default*: If you do not specify this parameter, the detailed information is output to the ptnName.wireCrossing file, where ptnName is the name of the partition. For master clone designs, three output files are created. These files can be used by the `add_partition_feedthrus` command for path based feedthrough insertions in master and clone designs. For example, when you specify, `gui_show_partition_wire_cross_over -out_file abc` The following files are generated: • abc.mc_connected_nets.txt: For all nets connected to master/clone partitions. • abc.mc_crossing_nets.txt: For all nets with wires over master/clone partitions (but not in the above file) and not intra partition nets. • abc.non_mc.txt: For all remaining nets which are not in above files and are not intra partition nets. **Note**: The default output file, abc.wireCrossing, is now only valid for non master/clone designs and has no significance for master and clone designs. *Data_type*: string, optional
partition_name	Specifies the name of the partition to show wire crossing. If you choose this option with other options, it must be the first specified when running this command. *Default*: If you do not specify this parameter, it will show all partition wire crossings. *Data_type*: string, optional

Example

The following command displays the routes crossing over partition, sheet17, and outputs the detail information into file, sheet17.crossing:

```
gui_show_partition_wire_cross_over sheet17 -out_file sheet17.crossing
```

The wire crossing report section lists the nets, their wire lengths, in micrometers, and the shape of the wire in relation to the partition.
For example, the following report segment is for a partition module named `sheet17`:

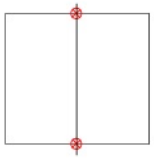
```
#########################################################
# Nets that cross partition module sheet17
# Box (335 335) (833 567)
```

```
# Format: Net <netName> <wireLength> <shape>
##########################################################

Net A 65 I
Net B 80 L
Net C 1050 T
Net D 132 X
...
```

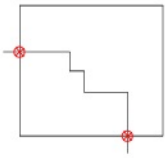
The first net in the report, A, has a wire length of 65 micrometers in an `I' shape, which indicates that the net crosses the partition on opposite sides, as follows:

```
Net A 65 I
```



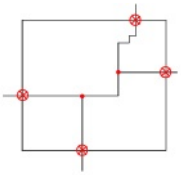
The second net in the report, B, has a wire length of 80 micrometers in an `L' shape, which indicates that the net crosses the partition on adjacent sides, as follows:

```
Net B 80 L
```



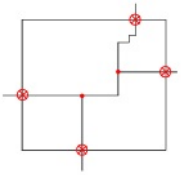
The third net in the report, C, has a wire length of 105 micrometers in an `T' shape, which indicates that the net crosses the partition on three sides, as follows:

```
Net C 105 T
```



The fourth net in the report, D, has a wire length of 132.30 micrometers in an `X' shape, which indicates that the net crosses the partition on all four sides, as follows:

```
Net D 132 X
```



Related Information

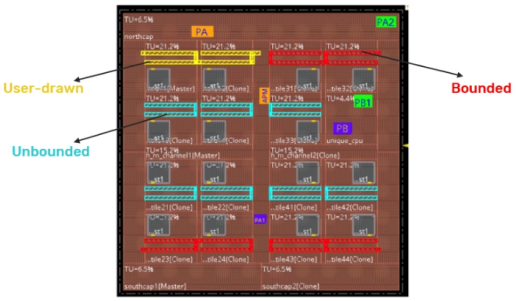
- `add_partition_feedthrus`
- Partitioning the Design chapter in the *Innovus Stylus Common UI User Guide*
  - Generating the Wire Crossing Report

highlight_sync_obj

```
highlight_sync_obj
[-help]
[-obj {all bus_guides}]
```

Adds highlight color on sync objects during master-clone-aware bus guide operations. The colors used for highlighting are based on different color schema categories – user-drawn, bounded, or unbounded. These categories are derived from the last sync command (`Clone_partition_obj -sync` or `set_db floorplan_auto_sync_master_clone busGuides`) used. This command makes it easy to distinguish between different bus guide objects in the design.

- User-drawn bus guides are highlighted in yellow.
- Bounded bus guides are highlighted in red. Nets in bounded bus guides can be determined during sync operation. If either (or both) the source or driver partition is a master-clone, the tool can determine the equivalent net during sync (in master clone chains).
  For example: Consider A1-B1-C1 and A2-B2-C2 chains where A/B/C1 A/B/C2 are master clones. A1/2 pin k is connected to net n1 (in A1) and n2 (in A2). As we can easily determine n1 and n2 in the chains, this is a case of bounded bus guides.
- Unbounded bus guides are highlighted in blue. Nets in unbounded bus guides cannot be determined during sync operation. If both source or driver partition are unique or (special case master and clone but their equivalent pin is floating), the tool cannot determine the net during sync (in master and clone chains).
  For example: Consider UA1-B1-UC1 and UA2-B2-UC2 chains where B1 and B2 are master clones. UA1/UC1/UA2/UC2 are all unique but UA1/UA2 pin may be different. Here since we cannot determine equivalent nets of chains, this is a case of unbounded bus guides.



Parameters

`-help`	Outputs a brief description that includes the type and default information for each `highlight_sync_obj` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man highlight_sync_obj`
`-obj {all bus_guides}`	Specifies the object to be highlighted. *Data_type*: (enum)+, optional

Example

- The following command highlights busguides based on different color schema categories
  `highlight_sync_obj -obj bus_guides`

Related Information

- Partitioning the Design chapter in the *Innovus Stylus Common UI User Guide*

legalize_pins

```
legalize_pins
[-help]
[-auto_pair]
[-ignore {pin_spacing pin_width pin_depth pin_layer pin_min_area pin_on_track pin_abutment pin_non_nbr_abutment clones}]
[-internal_pins]
[-keep_layer]
[-keep_order]
[-move_fixed_pins]
[-pins {pinName | pinNameList}]
[-partitions ptnName]
[-snap_to_boundary]
[-verbose]
[-partitions ptnName | {-pins_file fileName [-exclude_partitions ptnName]}]
[-pins_file fileName  | -exclude_pin_file fileName]
```

Moves a pin from its existing location in the following cases:

- The pin overlaps with any other object or pin. The objects considered are:The pin violates pin-to-pins spacing constraints
  - stripes
  - vias
  - feedthrough buffers
  - routing blockages
  - pin blockages
- The pin is on a non-preferred layer, a non-reserved layer, or the Metal1 layer
- The pin is not on the routing track
- The pin does not honor corner mask settings

The pin is assigned to the nearest legal location.

Note: If this command is run on a blackbox that has one or more pins with a single LEF port but with multiple shapes within the LEF port, the command will remove these multiple shapes and create a single shape. If this command is run on a blackbox that has one or more pins with multiple LEF ports, the first LEF port will be processed as per the command and the remaining LEF ports will remain unchanged.

The `legalize_pins` command supports nested partitions and corrects the pin position of illegal pins across hierarchies. It gives a warning if the first level pin is placed in second level partition or vice versa and legalizes them to the nearest track. It also supports pin overlapping across N levels for overlapping partition boundaries.

The l`egalize_pins` command honors net weights and accordingly legalizes pins. For example, if there are 2 or 3 pins that can be put into a slot, the `legalize_pins` command automatically looks at the net weight of these pins and assigns the slot according the net weight priority.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `legalize_pins` parameter. For a detailed description of the command and all of its parameters, use the man command: `man legalize_pins`
`-auto_pair`	Automatically calculates the channel width or distance between pins in a design which will be considered for pairing. *Data_type*: bool, optional
`-exclude_partitions ptnName`	Specifies the list of excluded partition for pin file. *Data_type*: string, optional
`-exclude_pin_file fileName`	
	Specifies pin-list to be excluded for partition in a file. *Data_type*: string, optional
`-ignore {pin_spacing pin_width pin_depth pin_layer pin_min_area pin_on_track pin_abutment pin_non_nbr_abutment clones}`	

	Legalizes by taking the category(ies) of ignore types into account. *Data_type*: (enum)+, optional
`-internal_pins`	Legalizes internal pins by moving them to the closest track on the preferred routing layer. *Data_type*: bool, optional
`-keep_layer`	Maintains the pin layer while legalizing the pins. If the pin is on a layer that is not on the allowed partition layer list then an error is reported and the pin is not legalized. *Data_type*: bool, optional
`-keep_order`	Maintains the pin order while legalizing pins irrespective of the layers. Initially, the pins are legalized within the range of original pin locations. However, if the `legalize_pins -keep_order` command is unable to find a legal location, `legalize_pins` will try to find a valid location that is outside the original range and give warning message. If `legalize_pins` fails to find a valid location, it reports an error. **Note**: The option is only applicable for pins on the fence. It does not apply to internal pins. *Data_type*: bool, optional
`-move_fixed_pins`	If you use this parameter, the specified pins with a Fixed status can also be moved. **Note**: Pins that are not specified but have a Fixed status are not moved. *Default*: If you do not specify this option, the specified pins with a Fixed status are not moved. *Data_type*: bool, optional
`-partitions` *ptnName*	Specifies the partition for which the pins will be checked. This parameter is required if the partition is not the top-level module. **Note**: The `legalize_pins` command can also resolve I/O pins from overlapping power stripes. In this case, *ptnName* specifies the block-level design name. *Data_type*: string, optional
`-pins {`*pinName* `|` *pinNameList*`}`	
	Specifies that the command should be run on the specified pin(s) only. *Data_type*: string, optional

`-pins_file` *fileName*	Specifies the pin list for a partition in a file. In the pin file, the pins and nets are specified in the following format:
	`Partition: PtnName/BlockName`
	Pin1
	Pin2
	..
	`Partition: PtnName/BlockName`
	Pin1
	Pin2
	..
	`Nets:`
	netA
	netB
	..
	Note: While specifying the pin list in the pin_file, it is necessary to give the “Partition” keyword followed by the name of the partition or block. If this is not specified, no operations are performed. You can specify multiple partitions in the same pin_file.
	The `pin_file` does not support buses or wild cards for pin names. If a pin specified in the list or a partition is not present in the design, a warning is displayed.
	Data_type: string, optional

`-snap_to_boundary`	Snaps internal partition pins to the boundary and legalizes them. **Note**: The `-snap_to_boundary` parameter does not work with `-internal_pins` as both work on internal pins and their results are different. *Data_type*: bool, optional

`-verbose`	Reports all the existing pin violations. *Data_type*: bool, optional

Example

The following command legalizes pins in same the metal as they were present and moves fixed pins if they are placed illegally:

```
legalize_pins -keep_layer -move_fixed_pins
```

The following command accepts a file that contains the list of pins for partition A and partition B and legalizes them.

```
legalize_pins -partitions {A B} -pins_file pinLst.txt
```

where, the *pinList.txt*

```
Partition: top
out1
out2
out3
Partition: A
in1
in2
in3
Partition: B
pina
pinb
pinc
```

Related Information

- `assign_partition_pins`
- `assign_io_pins`
- `check_pin_assignment`
- Partitioning the Design chapter in the *Innovus Stylus Common UI User Guide*
  - Validating Pin Placement

partition_floorplan Category Attributes

The root attributes in the `partition_floorplan` category are:

- partition_floorplan_export
- partition_floorplan_import

To set the value of any root attribute, use the `set_db` command:

```
set_db attribute_name value
```

To get the current value of any root attribute, use the `get_db` command:

```
get_db attribute_name
```

To get the current value of all attributes in the `partition_floorplan` category, use the following command:

```
get_db -category partition_floorplan
```

partition_floorplan_export

`partition_floorplan_export {macro | special_route | pin_constraint}`

Default: `special_route` (default data + special route is always saved)

Data_type: enums={macro special_route pin_constraint}_list, read/write

Specifies the objects to be exported. This is used by the `write_partitions -floorplan_only` command to save the data to the fp file.

Related Commands

- `write_partitions`
- `assemble_design`

partition_floorplan_import

`partition_floorplan_import {macro | special_route | pin_constraint}`

Default: By default, it brings only defaults objects

Data_type: enums={macro special_route pin_constraint}_list, read/write

Specifies the objects to be imported. This is used by the `assemble_design -floorplan_only` command to load the data from the fp file.

Related Commands

- `write_partitions`
- `assemble_design`

partition_push_network

```
partition_push_network
[-help]
[-exclude_partitions partition_name_list]
[-filter string]
[-inst_prefix inst_prefix_name]
[-logical_only]
[-net_prefix net_prefix_name]
[-out_file_logical_only_nets string]
[-output_nets_for_wire_distribution string]
[-partition_net_dir file_name]
{-nets net_name_list | -in_file file_name | -buffer_only  | -inst inst_name_list }
[-check_only [-out_file file_name]]
```

Pushes down a network (std cells and connected wires) logically and physically into overlapping partitions. The `partition_push_network` command:

- Logically pushes down a set of nets into overlapping partitions by doing all the necessary logical port creations.
- Physically pushes down the special wires of the nets into overlapping partitions and does all the necessary partition pin creations. The pushed wires look exactly the same as the original wires.
- Creates Verilog assign statements wherever required to ensure that the pushed logic is logically equivalent to the original logic.

Notes:

- The `partition_push_network` command does not support designs with master and clone partitions. In such scenarios, only masters will be seen and the overlapping wires and logic will be pushed down into masters only.
- The `partition_push_network` command takes care of new nets connected at top level nets automatically post committing partitions.
- The `partition_push_network` command supports partially routed nets. It does not require the wires of a net to be DRC/LVS clean. Wires need not connect to the instance-pin of a driver or loads. Even if a wire is in the 5u radius of a source-instance-pin or a sink-instance-pin, the tool considers the net for pushdown.

Parameters

`-buffer_only`	Specifies that only all buffers, and their connected nets, will be considered for pushed down. *Data_type*: bool, required
`-check_only`	Informs which nets are going to be pushed down. *Data_type*: bool, optional
`-exclude_partitions` *partition_name_list*	
	Specifies that the instances/nets within the specified partition(s) will not be pushed down. *Data_type*: string, optional
`-in_file` *fileName*	Specifies the name of the file containing the list of net names to be considered for push down. *Data_type*: string, required
`-filter` *string*	Specifies that only instances/nets whose name matches this specified regular expression will be pushed down. *Data_type*: string, optional
`-help`	Outputs a brief description that includes the type and default information for each `partition_push_network` parameter. For a detailed description of the command and all of its parameters, use the man command: `man partition_push_network`
`-inst_prefix` *inst_prefix_name*	Specifies a custom prefix for instances that are pushed down. *Data_type*: string, optional
`-inst` *inst_name_list*	Specifies that all instances, and their connected nets, will be considered for pushed down. *Data_type*: string, required
`-logical_only`	Logically pushes down network into overlapping partitions. *Data_type*: bool, optional
`-net_prefix` *net_prefix_name*	Specifies a custom prefix for nets that are pushed down. *Data_type*: string, optional
`-nets` *net_name_lst*	Specifies that list of net names, and their connected instances, will be considered for pushed down. *Data_type*: string, required
`-out_file` *file_name*	Specifies a file name in which net names, to be pushed down, will be mentioned. This parameter is used with `-check_only`. *Data_type*: string, optional
`-out_file_logical_only_nets` *string*	
	Specifies a file name in which net names are mentioned. This parameter is used with `-logical_only`. *Data_type*: string, optional
`-output_nets_for_wire_distribution` *string*	
	Outputs the user-specified nets so that they can be used later during the `commit_partitions` command. *Data_type*: string, optional
`-partition_net_dir` *file_name*	
	Specifies the directory that consists of the names of the pushed nets corresponding to each partition. *Data_type*: string, optional

Example

- The following command consider nets n1, n2 and n3 for push down.
  ```
  partition_push_network -nets {n1 n2 n3}
  ```
- The following command generates a file, push_down_nets.txt, containing the names of nets that would have been considered for push down.
  ```
  partition_push_network -nets [get_db nets .name] -check_only -out_file push_down_nets.txt
  ```
- The following command generates a file, push_down_nets.txt, containing the names of nets (connected to the specified instances)that would be considered for push down.
  ```
  partition_push_network -inst [get_db insts .name] -check_only -out_file push_down_nets.txt
  ```
- The following command considers all buffers for push down.
  ```
  partition_push_network -buffer_only
  ```
- The following command considers all buffers and their connected nets for push down which have their hierarchical name (TCL) regexp matching to KK.*.
  ```
  partition_push_network -buffer_only -filter KK.*
  ```
- The following command considers all buffers and their connected nets for push down. The pushed down buffers will have their base names starting with II
  ```
  partition_push_network -buffer_only -inst_prefix II
  ```
- `The following command` considers all buffers and their connected nets for push down. The pushed down buffers will have their base names starting with II, and the pushed down nets will have their base names starting with NN_.`partition_push_network -buffer_only -inst_prefix II -net_prefix NN`

Related Information

- `create_pushdown_eco`
- `commit_pushdown_eco`
- `commit_partitions`
- Partitioning the Design chapter in the *Innovus Stylus Common UI User Guide*
  - Partition Pushdown Replay Flow

partition_push_special_wires

```
partition_push_special_wires
[-help]
[-check_only]
[-nets list_of_net_names]
```

Creates Verilog ports (connections) in partitions for nets whose special wires overlap a partition, so that the special wire of the net gets pushed down when the `commit_partitions` command is used.

Note: For pushdown of special wires in master and clone partitions, it checks for the symmetry of the special wire shapes. For symmetrical shapes only, the `partition_push_special_wires` command creates one port in the Verilog module and establishes a logical connection between the new port and the net of special wire over all master and clone hinsts (reuse of logical port in master and clones).

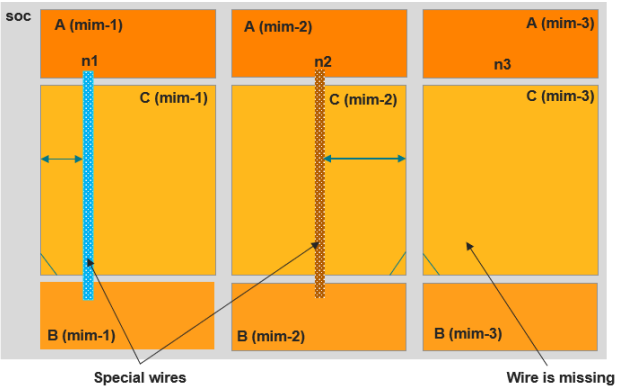
Note: If the original net was connected to a partition port of type inout, then an inout port is created inside the overlapping partition else only an input port is created.

Parameters

`-check_only`	When specified, the tool checks if the selected nets special-wires are symmetric over all hints of overlapping master clone cells or not. In master and clone partitions, it checks for the symmetry of nets and reports asymmetrical nets. When the `-check_only` parameter is not specified, the tool check the nets specified and creates logical ports only on valid nets. Data_type: bool, optional
`-help`	Outputs a brief description that includes the type and default information for each `partition_push_special_wires` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man partition_push_special_wires`
`-nets list_of_net_names`	Specifies the list of nets (Verilog nets) which have special wires to be pushed into blocks. *Data_type*: string, required

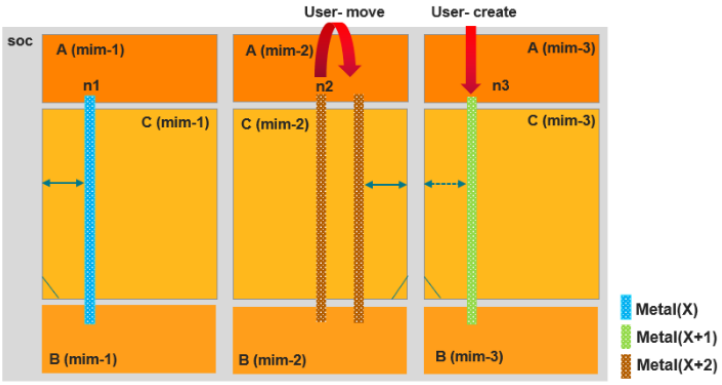
Example

- The following command checks the special wires of specified Verilog nets n1 and creates ports on valid nets.
  ```
  partition_push_special_wires -nets {n1}
  ```
- The following command checks if the special-wires of the specified nets are symmetric over all hints of overlapping master clone cells or not.
  ```
  partition_push_special_wires -nets {n1 n2} -check_only
  ```
- The following command checks if the special-wires of the specified nets are symmetric over all hints of overlapping master clone cells or not.
  ```
  partition_push_special_wires -nets {n1 n2} -check_only
  ```



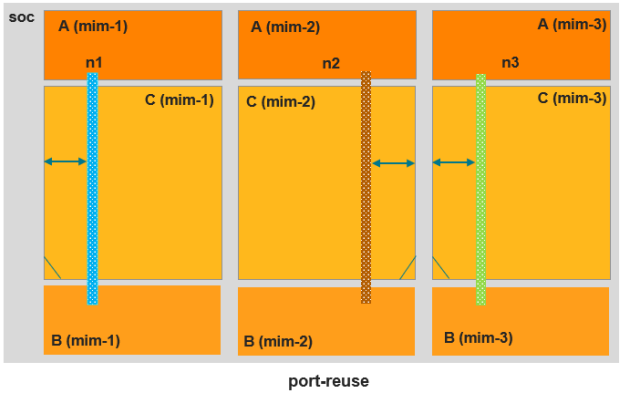

```
Report: C (mim-1) net n1 box missing in C mim-2
Report: C (mim-1) net n1 box missing in C mim-3
Report: C (mim-2) net n2 box missing in C mim-1
Report: C (mim-2) net n2 box missing in C mim-3
```

For pushdown, you must fix the wire asymmetry.



After the symmetry is fixed, the following command pushes down symmetrical nets:

```
partition_push_special_wires -nets {n1 n2}
```




```
Report: user Net n1, n2 and derived nets n3 pushdown, with one logical port
```

Related Information

- Partitioning the Design chapter in the *Innovus Stylus Common UI User Guide*

place_cells_at_center_for_feedthrus

```
place_cells_at_center_for_feedthrus
[-help]
[-insts list_of_selected_insts]
```

Places all the instances corresponding to a given partition at its center to facilitate feed-through insertion in the flow during the floorplanning stage. It quickly changes the design status to “placed” by placing all unplaced cells in the middle of the partition, so that feedthrough can be run.

Note: It places all the cells on the same location (overlapping).

Note: It can handle rectilinear partition fences and nested designs.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `place_cells_at_center_for_feedthrus` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man place_cells_at_center_for_feedthrus`
`-insts list_of_selected_insts`	Specifies the name of the instances that will be placed at the center of the partition. You can use wildcards (*?) with this parameter. For example, the following command places all unplaced cells that have ECO in their name to the center of the partition: `place_cells_at_center_for_feedthrus -insts *ECO*` *Data_type*: string, optional

Example

The following command changes the design status to “placed” by placing all unplaced cells in the middle of the partition, so that feedthrough can be run.

```
place_cells_at_center_for_feedthrus
```

Related Information

- `add_partition_feedthrus`
- `create_partition_feedthru`
- Partitioning the Design chapter in the *Innovus Stylus Common UI User Guide*
  - Inserting Feedthroughs

place_partition_clone

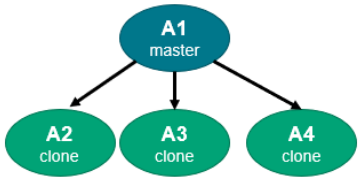
```
place_partition_clone
[-help]
-hinst partition_clone_hinst_name
-location {x y}
[-orient {r0 r80 mx my}]
[-skip_sync]
[-sync_objs {placeBlockage routeBlockage pinBlockage macros fences}]
```

Places the clones, set their orientation and syncs up the data with other hierarchical instances (master and clones) of the same module.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `place_partition_clone` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man place_partition_clone`
`-hinst partition_clone_hinst_name`	Specifies the name of partition clone hierarchical instance. *Data_type*: string, required
`-location {x y}`	Specifies location of the clone partition. *Data_type*: point, required
`-orient {r0 r80 mx my}`	Specifies orientation of the clone partition. The orientation can be one of the following: `r0, r180, mx, and my` *Data_type*: enum, optional
`-skip_sync`	When specified, the tool skips the sync-up of reference hinst (`-hinst`) objects with other hinsts of the same module. *Data_type*: bool, optional
`-sync_objs {placeBlockage routeBlockage pinBlockage macros fences}`	When specified, the tool does the sync-up of the specified objects from reference hinst with other hinsts of same module. The objects that can be synced up include placeBlockage, routeBlockage, pinBlockage, macros, and fences. *Data_type*: string, optional

Example




```
Hinst: A1 (master), A2 (clone), A3(clone), A4 (clone)
```

- The following commands places the clone A2 at location 2,5. If partition block has FP/def/route objs/data present, the command will sync the data from A2 to all other hinsts (A1 (master), A3 , A4 (clones)).
  ```
  place_partition_clones -hinst A2 -location {2 5}
  ```
- The following commands places the clone A3 at location 12,15 and changes the orientation to r180 (if not already R180). If partition block has FP/def/route objs/data present, the command will sync the data from A3 to all other hinsts (A1 (master), A2 , A4 (clones))
  ```
  place_partition_clones -hinst A3 -location {12 15} -orient r180
  ```
- The following commands places the master A1 at location 22,25. If partition block has FP/def/route objs/data present, the command will sync the data from A1 to all other hinsts (A2, A3, A4 (clones)).

```
place_partition_clones -hinst A1 -location {22 25}
```

- The following commands places the master A1 at location 22,25. If partition block has FP/def/route objs/data present, the command will not sync any data to other hinsts.

```
place_partition_clones -hinst A1 -location {22 25} -skip_sync
```

- The following commands places the clone A3 at location 12,15 and changes the orientation to r180 (if not already r80). If partition block has FP/def/route objs/data present, the command will not sync any data to other hinsts.

```
place_partition_clones -hinst A3 -location {12 15} -orient r180 -skip_sync
```

- The following commands places the clone A3 at location 12,15 and changes the orientation to r180 (if not already r180). If partition block has FP/def/route objs/data present, the command will sync the bus-guide data from A3 to all other hinsts (A1 (master), A2 , A4 (clones)).

```
place_partition_clones -hinst A3 -location {12 15} -orient r180 -sync_obj {bus_guide}
```

Related Information

- `clone_partition_obj`
- Partitioning the Design chapter in the *Innovus Stylus Common UI User Guide*

read_black_box_netlist

```
read_black_box_netlist
[-help]
netlist_file
```

Loads a netlist_file for a black box. You can use this command to incrementally load an updated netlist_file for a blackbox. You can use this command before converting a blackbox to a fence.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `read_black_box_netlist` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man read_black_box_netlist`
`netlist_file`	Specifies the netlist_file name. The netlist_file can contain more than one black box netlist_file, but it should not contain any non-blackbox netlist_file. *Data_type*: string, required

Example

The following command loads the netlist_file `netlist_file_v1`.

```
read_black_box_netlist netlist_file_v1
```

Related Information

- `create_blackbox`
- `delete_black_boxes`
- Partitioning the Design chapter in the *Innovus Stylus Common UI User Guide*
  - Specifying Partitions and Blackboxes
  - Blackbox Flow

read_hierarchical_def

```
read_hierarchical_def
[-help]
<def_files <...>>
[-add_blockage]
[-delete_blockages_over_partition_macro]
[-ignore_drcfill]
[-incremental_ilm_def]
[-rdl_def string]
[-rdl_orient {r0 r90 r180 r270 mx mx90 my my90}]
[-rdl_placement {x y}]
[-secondary {{suffix def_file_list}...}]
[-skip_fillers]
[-skip_nets] [-skip_pg] [-skip_routing string]
[-to_layer routing_layer_name]
[-top_cell_orient {r0 r90 r180 r270 mx mx90 my my90}]
[-top_cell_placement {x y}]
[-top_scope]
[-transform_file string]
[-use_top_def_die_area]
```

Merges DEF data from various blocks to a single chip database. The `read_hierarchical_def` command provides the following features:

- Takes the TOP DEF file and a set of block DEF files and merges them into a single chip database.
- The block and TOP Verilog files (or their DEF files) can be specified in any order. However, if there are empty Verilog module stubs of blocks in some of the parent Verilog files, ensure that the parent Verilog files are specified prior to the child Verilog files.
- There is no need to specify the LEF files for the blocks.
- During the course of this command, TOP rows will be cut around the block boundary, and all block rows will be brought back as is.
- Partition definitions are created for each of the assembled block as per the DIE AREA statement in the block DEF.
- The wire at the block boundaries will be stitched with the electrically connected net's wires at the TOP.
- All orientations of blocks are supported for assembly. Repeated blocks are also supported. Non-R0 master is also supported.

Parameters

`-add_blockage`	Specifies that the netlist is a reduced one and converts the undefined instances into blockages. *Data_type*: bool, optional
def_files <...>	Specifies DEF files to be merged. *Data_type*: list, required
`-delete_blockages_over_partition_macro`	
	Deletes top design placement and routing blockages overlapping partitions. *Data_type*: bool, optional
`-help`	Outputs a brief description that includes the type and default information for each `read_hierarchical_def` parameter. For a detailed description of the command and all of its parameters, use the man command: `man read_hierarchical_def`
`-ignore_drcfill`	Ignores the special routing with DRCFILL shape attribute. *Data_type*: bool, optional
`-incremental_ilm_def`	Incrementally flatten the ILM def. *Data_type*: bool, optional
`-rdl_def string`	Specifies RDL def file. *Data_type*: string, optional
`-rdl_orient {r0 r90 r180 r270 mx mx90 my my90}`	

	Specifies orientation for RDL def. *Data_type*: enum, optional
`-rdl_placement {x y}`	Specifies offset of RDL def. *Data_type*: point, optional
`-secondary {{suffix def_file_list}...}`	
	Specifies the secondary DEF file(s) to load. *Data_type*: string, optional
`-skip_fillers`	Skip merging filler and decap cells. *Data_type*: bool, optional
`-skip_nets`	Skips merging signal nets section. *Data_type*: bool, optional
`-skip_pg`	Skips merging routing on pg nets. *Data_type*: bool, optional
`-skip_routing {[pg|signal] [regular|special] [high_layer_routing]}...`	
	Skips merging routing based on the settings specified. This setting can be any combination of the three optional keywords from the following set: • `[pg|signal]`: Specifies the type of the net on which the routing to be skipped. pg – pg net, signal – signal net, both if not specified. • `[regular|special]`: Specifies the type of routing to be skipped. regular – regular routing, special – special routing, both if not specified. • `[high_layer_routing]`: Specifies that only the routing on high layers to be skipped, where the layers are calculated automatically. If not specified, the routing on all layers are skipped. The layers can be specified with `-to_layer` option. In that case, the routing on the layers from top to the specified layer are skipped. **Note:** You must specify the `-to_layer` parameter explicitly if `high_layer_routing` is specified for `-skip_routing`. Otherwise, `high_layer_routing` has no effect. *Data_type*: string, optional
`-to_layer routing_layer_name`	Skips the routing of layers from top to the specified layer (inclusive) when the `high_layer_routing` option is specified for the `-skip_routing` parameter. *Data_type*: (layer), optional
`-top_cell_orient {r0 r90 r180 r270 mx mx90 my my90}`	
	Specifies orientation for topcell. *Data_type*: enum, optional
`-top_cell_placement {x y}`	Specifies placement for topcell. *Data_type*: point, optional
`-top_scope`	Merges routing on top scope nets only. *Data_type*: bool, optional
`-transform_file string`	Reports the final x and y offsets and orientation in the top-level of each LEF block that got replaced by DEF. *Data_type*: string, optional
`-use_top_def_die_area`	Enables the use of TOP DEF DIE AREA. *Data_type*: bool, optional

Example

- The following command merges DEF data from `top.def.gz` and `blockA.def.gz` to a single chip database.

```
read_hierarchical_def top.def.gz blockA.def.gz
```

Related Information

- Partitioning the Design chapter in the *Innovus Stylus Common UI User Guide*

read_partition_pins

```
read_partition_pins
[-help]
[-report_file string]
[-snap_modified_to_boundary]
{-partition string | -all}
{-partition_file string| -def_file string}
```

Loads either a partition floorplan file (from a saved partition design session) or a DEF file, and then displays the preassigned I/O pins for the partition. This command must be run for each partition in the design. You can use this command on a partition design after loading the top level floorplan and after running/loading placement and early global route data.

Note: The `read_partition_pins` command can load partition files that are saved using the `write_floorplan` command.

Parameters

`-all`	Specifies that pin positions for all the partitions should be loaded. *Data_type*: bool, optional
`-def_file string`	Specifies the full pathname of the partition def file. **Note:** You can specify a DEF file in the gzip format (with the .gz extension) also. *Data_type*: string, optional
`-help`	Outputs a brief description that includes the type and default information for each `read_partition_pins` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man read_partition_pins`
`{-partition_file string | -def_file string }`	
	Specifies the full path name of the partition floorplan file. *Data_type*: string, optional
`-partition string`	Specifies the name of the partition. *Data_type*: string, optional
`-report_file string`	Specifies the optional report filename that contains any pins deleted or newly added. *Data_type*: string, optional
`-snap_modified_to_boundary`	Snaps pins to the modified partition boundary. The pin is unplaced if it cannot be snapped to edge at same location in sam the `read_partition_pins` command does not snap it to the track and you need to run the `legalize_pins` command to snap *Data_type*: bool, optional

Example

The following command loads pin information from file `EUG.fp.ptn` for partition `EUG`

```
read_partition_pins -partition EUG -partition_file EUG.fp.ptn
```

Related Information

- `write_floorplan`
- `assign_partition_pins`
- `reset_partition_pins`
- `write_partition_pins`
- Partitioning the Design chapter in the *Innovus Stylus Common UI User Guide*
  - Assigning Partition and Blackbox Pins
    - ECO Pin Assignment

report_partitions

```
report_partitions
[-help]
[top_cell]
```

Reports the partitions and clones in a correct tree like format.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `report_partitions` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man report_partitions`
top_cell	Specifies the top cell to print its cell partition tree. *Data_type*: string, optional

Example

The following command prints the partitions and clones in the design in a tree like format.

```
report_partitions
```

```
Format: partitionName, MasterOrClone, Orient, PartitionOrBlackBox (CommittedOrUncommited, Level, BottomUpLevel, In/Active): InstOrHInstName

- top (design name)
- top_partition(0x7fec92643638), Single, R0, Partition (Uncommitted,1,2,Active): T1
- sh_act_3(0x7fec92643310), Single, R0, Partition (Uncommitted,2,0,Active): T1/S2
- sh_act_2(0x7fec92642fe8), Single, R0, Partition (Uncommitted,2,0,Active): T1/S3
- sh_act_1(0x7fec92642cc0), Single, R0, Partition (Uncommitted,2,0,Active): T1/S4
- sh_act(0x7fec92642998), Single, R0, Partition (Uncommitted,2,0,Active): T1/S1
- dpu(0x7fec94c40bc0), Clone, R0, Partition (Uncommitted,2,1,Active): T1/P2
- scl(0x7fec94c405c0), Clone, R0, Partition (Uncommitted,3,0,Active): T1/P2/c15
- scl(0x7fec94c40560), Clone, R0, Partition (Uncommitted,3,0,Active): T1/P2/c14
- scl(0x7fec94c40500), Clone, R0, Partition (Uncommitted,3,0,Active): T1/P2/c13
- scl(0x7fec94c404a0), Clone, R0, Partition (Uncommitted,3,0,Active): T1/P2/c12
- scl(0x7fec94c40440), Clone, R0, Partition (Uncommitted,3,0,Active): T1/P2/c11
- scl(0x7fec94c403e0), Clone, R0, Partition (Uncommitted,3,0,Active): T1/P2/c10
- scl(0x7fec94c40380), Clone, R0, Partition (Uncommitted,3,0,Active): T1/P2/c9
- scl(0x7fec94c40320), Clone, R0, Partition (Uncommitted,3,0,Active): T1/P2/c8
- scl(0x7fec94c402c0), Clone, R0, Partition (Uncommitted,3,0,Active): T1/P2/c7
- scl(0x7fec94c40260), Clone, R0, Partition (Uncommitted,3,0,Active): T1/P2/c6
- scl(0x7fec94c40200), Clone, R0, Partition (Uncommitted,3,0,Active): T1/P2/c5
- scl(0x7fec94c401a0), Clone, R0, Partition (Uncommitted,3,0,Active): T1/P2/c4
- scl(0x7fec94c40140), Clone, R0, Partition (Uncommitted,3,0,Active): T1/P2/c3
- scl(0x7fec94c400e0), Clone, R0, Partition (Uncommitted,3,0,Active): T1/P2/c2
- scl(0x7fec94c40080), Clone, R0, Partition (Uncommitted,3,0,Active): T1/P2/c1
- scl(0x7fec94c40020), Clone, R0, Partition (Uncommitted,3,0,Active): T1/P2/m1
- dpu(0x7fec92642670), Master, R0, Partition (Uncommitted,2,1,Active): T1/P1
- scl(0x7fec94c40b60), Clone, R0, Partition (Uncommitted,3,0,Active): T1/P1/c15
- scl(0x7fec94c40b00), Clone, R0, Partition (Uncommitted,3,0,Active): T1/P1/c14
- scl(0x7fec94c40aa0), Clone, R0, Partition (Uncommitted,3,0,Active): T1/P1/c13
- scl(0x7fec94c40a40), Clone, R0, Partition (Uncommitted,3,0,Active): T1/P1/c12
- scl(0x7fec94c409e0), Clone, R0, Partition (Uncommitted,3,0,Active): T1/P1/c11
- scl(0x7fec94c40980), Clone, R0, Partition (Uncommitted,3,0,Active): T1/P1/c10
- scl(0x7fec94c40920), Clone, R0, Partition (Uncommitted,3,0,Active): T1/P1/c9
- scl(0x7fec94c408c0), Clone, R0, Partition (Uncommitted,3,0,Active): T1/P1/c8
- scl(0x7fec94c40860), Clone, R0, Partition (Uncommitted,3,0,Active): T1/P1/c7
- scl(0x7fec94c40800), Clone, R0, Partition (Uncommitted,3,0,Active): T1/P1/c6
- scl(0x7fec94c407a0), Clone, R0, Partition (Uncommitted,3,0,Active): T1/P1/c5
- scl(0x7fec94c40740), Clone, R0, Partition (Uncommitted,3,0,Active): T1/P1/c4
- scl(0x7fec94c406e0), Clone, R0, Partition (Uncommitted,3,0,Active): T1/P1/c3
- scl(0x7fec94c40680), Clone, R0, Partition (Uncommitted,3,0,Active): T1/P1/c2
- scl(0x7fec94c40620), Clone, R0, Partition (Uncommitted,3,0,Active): T1/P1/c1
- scl(0x7fec92642348), Master, R0, Partition (Uncommitted,3,0,Active): T1/P1/m1
```

Related Information

- Partitioning the Design chapter in the *Innovus Stylus Common UI User Guide*

report_pin_assignment_statistics

```
report_pin_assignment_statistics
[-help]
[-max_fanout fan_out_count]
[-max_register_fanout register_fan_out_count]
[-out_file pin_qor_file_name]
[-hinst hinst_name_list | -partition ptn_name_list]
```

Generates the pin QoR data for a specified partition(s).

Note:

- If a net is not routed, then, manhattan distance between source-sink pairs is be used to replace corresponding wire-length distance.
- For floating pins (from outside) of a partition,the pin is always assumed to be within the timing cone box.
- Pin QoR statistics is not generated for blackbox partitions

Parameters

`-help`	Outputs a brief description that includes type and default information for each `report_pin_assignment_statistics` parameter. For a detailed description of the command and all of its parameters, use the man command: `man report_pin_assignment_statistics`
`-hinst hinst_name_list`	Generates the pin QoR metrics for all pins of the specified hierarchical instance name or list of instance names. *Default*: All partition and clone hierarchical instances. *Data_type*: string, optional
`-max_fanout fan_out_count`	Specifies the maximum fanout count. The QoR report is not generated for a pin if the number of instances in its timing cone fanout exceeds this maximum fanout count, *Default*: 1000 *Data_type*: string, optional
`-max_register_fanout register_fan_out_count`	
	Specifies the maximum register fanout count. The QoR report is not generated for a pin if the number of registers in its timing cone fanout exceeds this maximum fanout count. *Default*: 50 *Data_type*: string, optional
`-out_file pin_qor_file_name`	Specifies an output report file for the pin QoR data. *Default*: `./pin_qor.txt` *Data_type*: string, optional
`-partition ptn_name_list`	Generates the pin QoR metrics for all pins of the partition or list of partitions. *Default*: All partition and clones. *Data_type*: string, optional

Example

- The following command creates a `pin_qor.txt` report file:
  ```
  report_pin_assignment_statistics
  ```
- The following command generates a reference pin QoR file named `pin_qor_reference.txt`.
  ```
  report_pin_assignment_statistics -out_file pin_qor_reference.txt
  ```
- The following command generates a target pin QoR file named `pin_qor_target.txt`.
  ```
  report_pin_assignment_statistics -out_file pin_qor_target.txt
  ```

Related Information

- `assign_partition_pins`
- `assign_io_pins`
- `compare_pin_assignment_statistics`
- Partitioning the Design chapter in the *Innovus Stylus Common UI User Guide*

report_two_pin_chain

```
report_two_pin_chain
[-help]
[-reset  |
[[-out_file file_name] [-through string]
[-no_overwrite ]
[-show_all ]
[-strict ]
[-tolerance value]]]
```

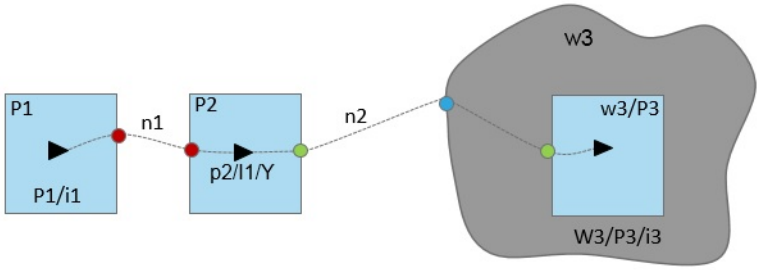
Reports two pin chains for nets, instterms, and hinstterms. This command can traverse through buffers, inverters, and assign-statements and report the complete two pin chain. You can specify any element of the chain to get the same results. The `report_two_pin_chain` command is used after pin assignment to analyse results and debug issues especially in cases where the expectation to align all partition pins across a two-pin chain is not met.

Parameter

`-help`	Outputs a brief description that includes type and default information for each `report_two_pin_chain` parameter. For a detailed description of the command and all of its parameters, use the man command: `man report_two_pin_chain`
`-no_overwrite`	When specified the report continues to display the previous 2-pin-chains also. *Default*: false *Data_type*: bool, optional
`-out_file file_name`	Specifies the file in which the 2-pin nets are traced and dumped. *Default*: stdout *Data_type*: string, optional
`-reset`	Removes GUI markers and GUI objects, created during previous invocations. No two pin chain reporting. *Default*: false *Data_type*: bool, optional
`-show_all`	Displays the intermediate instances of the chain also. *Default*: false *Data_type*: bool, optional
`-strict`	Strictly displays only pure two pin chains even at module port connections. *Default*: false *Data_type*: bool, optional
`-through string`	Specifies the list of names of net, instterms, hinstterms for which 2-pin chains are traced and dumped. *Data_type*: string, optional
`-tolerance value`	Specifies the tolerance range. This specified difference in ordinates treated as still aligned. *Default*: 1.0 microns *Data_type*: float, optional

Example

- In the following diagram P1, P2, W3/P3 are partitions and W3 is a non-partition verilog module.
  It shows a pure two pin connection where red and green pins aligned together in one chain.



  The following command reports this two pin chain passing through p2/I1/Y:

```
report_two_pin_chain -through p2/I1/Y
```

- The following command reports the two pin chains passing through `DTMF_INST/TDSP_CORE_INST/port_pad_data_out[13]`:

```
report_two_pin_chain -through DTMF_INST/TDSP_CORE_INST/port_pad_data_out[13]
```

```
Report:
2-pin-chain through: DTMF_INST/TDSP_CORE_INST/port_pad_data_out[13] ...
chain terminator: other pin of buffer is multi-fanout
net: DTMF_INST/TDSP_CORE_INST/port_data_in[13]
buffer-term                    :output :placed      : DTMF_INST/TDSP_CORE_INST/TDSP_CORE_GLUE_INST/i_9686/Y
module-port                    :output :unplaced    : DTMF_INST/TDSP_CORE_INST/TDSP_CORE_GLUE_INST/port_data_in[13]
module-port                    :input  :unplaced    : DTMF_INST/TDSP_CORE_INST/PORT_BUS_MACH_INST/data_in[13]
buffer-term                    :input  :placed      : DTMF_INST/TDSP_CORE_INST/PORT_BUS_MACH_INST/FE_RC_7_0/A
net: DTMF_INST/FE_FEEDX_NET_D__tdsp_core__0_tdsp_portO_13_
buffer-term                    :output :placed      : DTMF_INST/TDSP_CORE_INST/PORT_BUS_MACH_INST/FE_RC_7_0/Y
module-port                    :output :unplaced    : DTMF_INST/TDSP_CORE_INST/PORT_BUS_MACH_INST/pad_data_out[13]
partition-pin                  :output :placed      : DTMF_INST/TDSP_CORE_INST/port_pad_data_out[13]
partition-pin                  :input  :placed      : DTMF_INST/RESULTS_CONV_INST/FE_FEEDX_NET_D__tdsp_core__0_tdsp_portO_13_
buffer-term                    :input  :placed      : DTMF_INST/RESULTS_CONV_INST/FE_OFC28_FE_FEEDX_NET_D__tdsp_core__0_tdsp_portO_13_/A
net: tdsp_portO[13]
buffer-term                    :output :placed      : DTMF_INST/RESULTS_CONV_INST/FE_OFC28_FE_FEEDX_NET_D__tdsp_core__0_tdsp_portO_13_/Y
partition-pin                  :output :placed      : DTMF_INST/RESULTS_CONV_INST/FE_FEEDX_P_tdsp_portO_13_
module-port                    :output :unplaced    : DTMF_INST/port_pad_data_out[13]
module-port                    :input  :unplaced    : IOPADS_INST/tdsp_portO[13]
buffer-term                    :input  :placed      : IOPADS_INST/Ptdspop13/I
net: port_pad_data_out[13]
buffer-term                    :output :placed      : IOPADS_INST/Ptdspop13/PAD
module-port                    :output :unplaced    : IOPADS_INST/tdsp_port_out[13]
chip-io                        :output :fixed       : port_pad_data_out[13]
chain terminator: chip-io
```

- The following command reports the two pin chains passing through `F2/D2/A/out` and writes it to `report.txt` file:

```
report_two_pin_chain -through F2/D2/A/out -out_file report.txt
```

Related Information

- `assign_partition_pins`
- Partitioning the Design chapter in the *Innovus Stylus Common UI User Guide*

report_unaligned_nets

```
report_unaligned_nets
[-help]
[-align_distance distance]
[-align_feedthru_pins]
[-block_pins_within_distance distance]
[-ignore_length  length]
[-net_box_deviation]
[-no_highlight]
[-out_file filename]
[-report_all_within_channel_width width]
[-statistics]
[-verbose]
[[ -partition_to_partition {aligned | unaligned | layermismatch | sameptn  |nonnbr  | multifanout | all | unplaced | none} ]
[-top_to_partition {aligned | unaligned | layermismatch | sameptn  |nonnbr  | multifanout | all | unplaced | none} ]]
[-pin_file filename | -exclude_pin_file filename ]
```

Report the names of the nets where the pins are not aligned for the following cases:

- nets that have two pins on two different partitions and/or clones and are not connected to top level hard-macro, standard cell, or I/O.
- nets that have two pins that are connected to the same partition.
- nets that have one pin connected to top level hard-macro, standard cell, or I/O and the other pin connected to a partition or a clone.

Notes:

- The nets reported by this command are also highlighted in the artwork window.
- The `assign_partition_pins` command internally calls the `report_unaligned_nets` command to generate the summary report at the command line.
- The `report_unaligned_nets` command measures the average misalignment with standard deviation for multi-partition nets.
- The `report_unaligned_nets` command detects and reports pins that are placed on non-facing edges.

Parameters

`-align_distance` *distance*	Specifies the distance, in microns, up to which the pins are assumed to be aligned. With this parameter specified, the `report_unaligned_nets` command assumes that the pins are aligned within the specified distance even though they are not physically aligned. *Data_type*: float, optional
`-align_feedthru_pins`	Reports the deviation of partition pins from chain's source-target box. *Data_type*: bool, optional
`-block_pins_within_distance` *distance*	Reports nets connected to macro-pin which are within specified distance from nearest partition boundary. *Default*: 3 microns *Data_type*: float, optional

`-exclude_pin_file` *filename*	Specifies the pin list to be excluded for a partition in a file. In the pin file, the pins and nets are specified in the following format: `Partition:` *PtnName/BlockName* *Pin1* *Pin2* .. *PinN* `Partition:` *PtnName/BlockName* *Pin1* *Pin2* .. *PinN* `Nets:` *netA* *netB* .. *netN* **Note:** While specifying the pin list in the pin_file, it is necessary to give the “Partition” keyword followed by the name of the partition or block. If this is not specified, no operations are performed. You can specify multiple partitions in the same pin_file. The pin_file does not support buses or wild cards for pin names. If a pin specified in the list or a partition is not present in the design, a warning is displayed. *Data_type*: string, optional
`-help`	Outputs a brief description that includes type and default information for each `report_unaligned_nets` parameter. For a detailed description of the command and all of its parameters, use the man command: `man report_unaligned_nets.`
`-ignore_length` *length*	Specifies the length, in microns. It ignores the nets from checking unalignment whose length is greater the specified length value. *Default*: 5000 These ignored nets will be reported in the ignored category in the output report file. *Data_type*: float, optional
`-net_box_deviation`	Report the deviation of partition pins from the connectivity box of the partition. *Data_type*: bool, optional
`-no_highlight`	Disables the highlight capability. *Data_type*: bool, optional
`-out_file` *fileName*	Specifies an output report file. **Note**: Default name for report file is `<designName>_unaligned.rpt` *Data_type*: string, optional
`-partition_to_partition {aligned | unaligned | layermismatch | sameptn |nonnbr | multifanout | all | unplaced | none}`	

	Reports nets that have two pins on two different partitions and/or clones and are not connected to top level hard-macro, standard cell, or I/O. • `aligned`: Reports nets that have aligned pins. • `all`: Reports all categories of unaligned pins. • `layerMismatch`: Reports nets that have pins at an aligned location but different layers. • `multiFanout`: Reports nets that have multiple fanouts • `none`: Disables reporting of nets. • `nonNbr`: Reports nets that have two pins on two different partitions that are not adjacent. • `samePtn`: Reports nets that have two pins that are connected to the same partition. • `unaligned`: Reports nets that have unaligned pins. • `unplaced`: Reports nets that have unplaced pins. *Data_type*: (enum)+, optional
`-pin_file` *filename*	Specifies the pin list for a partition in a file. In the pin file, the pins and nets are specified in the following format: `Partition:` *PtnName/BlockName* *Pin1* *Pin2* .. *PinN* `Partition:` *PtnName/BlockName* *Pin1* *Pin2* .. *PinN* `Nets:` *netA* *netB* .. *netN* **Note:** While specifying the pin list in the pin_file, it is necessary to give the “Partition” keyword followed by the name of the partition or block. If this is not specified, no operations are performed. You can specify multiple partitions in the same pin_file. The pin_file does not support buses or wild cards for pin names. If a pin specified in the list or a partition is not present in the design, a warning is displayed. *Data_type*: string, optional
`-report_all_within_channel_width` *width*	Reports all the unaligned nets within the specified channel width even if the `-ignore_length` parameter has been specified. *Data_type*: float, optional
`-statistics`	Reports average misalignment with standard deviation and layer difference count for aligned, layer mismatched, and unaligned ptnToPtn 2-pin nets of partitions. It is also used for reporting pin QOR. *Data_type*: bool, optional
`-top_to_partition {aligned | unaligned | layermismatch | sameptn |nonnbr | multifanout | all | unplaced | none}`	

	Reports nets that have one pin connected to top level hard-macro, standard cell, or I/O and the other pin connected to a partition or a clone. • `aligned`: Reports nets that have aligned pins. • `all`: Reports all categories of unaligned pins. • `layerMismatch`: Reports nets that have pins at an aligned location but different layers. • `multiFanout`: Reports nets that have multiple fanouts • `none`: Disables reporting of nets. • `nonNbr`: Reports nets that have two pins on two different partitions that are not adjacent. • `samePtn`: Reports nets that have two pins that are connected to the same partition. • `unaligned`: Reports nets that have unaligned pins. • `unplaced`: Reports nets that have unplaced pins. *Data_type*: (enum)+, optional
`-verbose`	Reports all categories in log file. By default, this parameter is not enabled and only the nets that have unaligned pins are reported in the log file. **Note**: There will be no change in the report file with use of verbose parameter. *Data_type*: bool, optional

The following are some of the attributes specified in the output report file to report the names of nets where the pins are not aligned:

- `layer mismatch`: Specifies nets that have pins at an aligned location but different layers.
- `non-neighbor`: Specifies nets that have two pins on two different partitions that are not adjacent.
- `ptnToPtn`: Specifies nets that have two pins on two different partitions and/or clones.
- `topToPtn`: Specifies nets that have one pin connected to top level hard-macro, standard cell, or I/O and the other pin connected to a partition or a clone.
- `unAligned`: Specifies nets that have unaligned pins.

Example

The following command creates an a.out report file:

```
report_unaligned_nets rptFile a.out -no_highlight -partition_to_partition all -top_to_partition all
```

A summary of the reported unaligned nets displays on the Innovus console and in the Innovus log file.
The following is a sample of the resulting a.out report file.

```
############################################################
#  Generated by:      Cadence Innovus
#  OS:                Linux x86_64(Host ID nofclo110)
#  Generated on:      Wed Feb 27 16:50:02 2015
#  Design:            NESTED_TOP
#  Command:           report_unaligned_nets -partition_to_partition all -top_to_partition all -rptFi...
############################################################
====================
Net Specific report:
====================
pin_i_TOP_A0__A1__A2forPorts_BB_[3] | nested-2-pin | aligned 3/4
  pin_i_A1__A2forPorts_BB_[3] (I_K2/I_K2) | pin_i_A0__A1__A2forPorts_BB_[3] (I_K2) | ptnToPtn | aligned
  pin_i_A2forPorts_BB_[3] (I_K2/I_K2/I_K2) | pin_i_A1__A2forPorts_BB_[3] (I_K2/I_K2) | ptnToPtn | aligned
  input_bus[3] (I_K2/I_K2/I_K2/I_A2_K4/BBOX) | pin_i_A2forPorts_BB_[3] (I_K2/I_K2/I_K2) | ptnToPtn | aligned
  pin_i_A0__A1__A2forPorts_BB_[3] (I_K2) | pin_i_TOP_A0__A1__A2forPorts_BB_[3] (NESTED_TOP) | topToPtn | unaligned
pin_i_TOP_A0__A1__A2forPorts_BB_[2] | nested-2-pin | aligned 1/4
  pin_i_A2forPorts_BB_[2] (I_K2/I_K2/I_K2) | pin_i_A1__A2forPorts_BB_[2] (I_K2/I_K2) | ptnToPtn | aligned
  pin_i_A0__A1__A2forPorts_BB_[2] (I_K2) | pin_i_TOP_A0__A1__A2forPorts_BB_[2] (NESTED_TOP) | topToPtn | unaligned
  pin_i_A1__A2forPorts_BB_[2] (I_K2/I_K2) | pin_i_A0__A1__A2forPorts_BB_[2] (I_K2) | ptnToPtn | unaligned
  input_bus[2] (I_K2/I_K2/I_K2/I_A2_K4/BBOX) | pin_i_A2forPorts_BB_[2] (I_K2/I_K2/I_K2) | ptnToPtn | unaligned
...
...
I_K3/net15_OP_B1_0_And_IP_B1_2_SC[3] | 2-pin | aligned 0/1
```

```
  pin_o_B1_0_III_SC_[0] (I_K3/I_K4) | pin_i_B1_2_SC_[0] (I_K3/I_K2) | ptnToPtn | nonNbr
I_K3/net16_OP_B1_0_And_IP_B1_2_SC | 2-pin | aligned 0/1
  pin_o_B1_0_III_SC_4 (I_K3/I_K4) | pin_i_B1_2_SC_4 (I_K3/I_K2) | ptnToPtn | nonNbr

==================
Net Summary report:
==================
There are 54 ptnToPtn 2-pin net(s)
There are 0 topToPtn 2-pin net(s)
There are 200 nested ptnToPtn 2-pin net(s) with multiple partition pins between source and sink
There are 34 nested topToPtn 2-pin net(s) with multiple partition pins between source and sink

================================================================================
                Summary of pin pairs on above nets
================================================================================
|aligned|unaligned|layerMismatch|samePtn|nonNeighbor|multiFanout|unplaced|ignored|Total
================================================================================
topToPtn |  11 |  22 |   1 |  0 |  0 |  0 |  0 |  0 |  34
ptnToPtn | 590 | 220 |  16 |  0 | 58 |  0 |  0 |  0 | 884
================================================================================
Total    | 601 | 242 |  17 |  0 | 58 |  0 |  0 |  0 | 918
================================================================================

================================================================================
                 Level based partition net summary
================================================================================
 |aligned|unaligned|layerMismatch|samePtn|nonNeighbor|multiFanout|unplaced|ignored|Total|
================================================================================
NESTED_TOP (level-0)    |121 | 30 |  1 |  0 |  0 |  0 |  0 | 0 | 152 |
I_K2 (level-1)          |161 | 18 |  7 |  0 |  0 |  0 |  0 | 0 | 186 |
I_K3 (level-1)          | 52 | 42 |  0 |  0 | 58 |  0 |  0 | 0 | 152 |
I_K2/I_K2 (level-2)     |184 | 35 |  1 |  0 |  0 |  0 |  0 | 0 | 220 |
I_K3/I_K2 (level-2)     | 10 |  6 |  0 |  0 |  0 |  0 |  0 | 0 |  16 |
I_K3/I_K3 (level-2)     | 16 | 16 |  0 |  0 |  0 |  0 |  0 | 0 |  32 |
I_K3/I_K4 (level-2)     | 12 | 32 |  4 |  0 |  0 |  0 |  0 | 0 |  48 |
I_K2/I_K2/I_K2 (level-3)| 45 | 63 |  4 |  0 |  0 |  0 |  0 | 0 | 112 |
================================================================================
```

Related Information

- assign_partition_pins
- Partitioning the Design chapter in the *Innovus Stylus Common UI User Guide*

reset_partition_pins

```
reset_partition_pins
[-help]
-partition partitionName
{ -in_file floorplanFileName }
```

Removes the loading of partition pins by the `read_partition_pins` command. This is used to remove the pre-assignment of pins that were previously reassigned by the loading.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `reset_partition_pins` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man reset_partition_pins`
`-in_file floorplanFileName`	Specifies the full path name of the partition floorplan file. *Data_type*: string, optional
`-partition partitionName`	Specifies the name of the partition. *Data_type*: string, required

Example

The following command unloads pin positions for partition `EUG` from file `EUG.fp.ptn`:

```
reset_partition_pins -partition EUG -in_file EUG.fp.ptn
```

Related Information

- `assign_partition_pins`
- `read_partition_pins`
- `write_partition_pins`
- Partitioning the Design chapter in the *Innovus Stylus Common UI User Guide*
  - Assigning Partition and Blackbox Pins
    - ECO Pin Assignment

reset_pin_constraint

```
reset_pin_constraint
[-help]
{[-global | -cell cell_name | -all_user ] [-global | -pins pin_name_list | -area {x1 y1 x2 y2} | -all_area | -side side_names]
[-global | -layer ] [-global | -location | -edge | -corner corner_number]
[[-pins pin_name_list [-location ] [-edge ]] |
[-corner_to_pin_distance [-corner corner_number]]]
[{-pins pin_name_list | -pin_template {layer_id_or_name_list}} -width ]
[{-pins pin_name_list | -pin_template {layer_id_or_name_list}} -depth ]
[[-layer [-side side_names]] | -pin_template {layer_id_or_name_list}]
[-spacing [-area {x1 y1 x2 y2}]
[-all_area ] [-side side_names]] }
[-depth [-special_wire ]] [-all_user ]
```

Removes the constraints for a partition pin. You can use this command after setting the pin's constraints with the `set_pin_constraint` command.

Parameters

`-all_area`	Specifies that spacing constraints on all areas should be removed. *Data_type*: bool, optional
`-all_user`	Specifies that all user specified constraints should be removed from all levels (`-global`, `-cell`, and `-pins` level). *Data_type*: bool, optional
`-area {x1 y1 x2 y2}`	Specifies that the area overlapping with partition for which the spacing constraints have been specified should be removed. *Data_type*: box, optional
`-cell cell_name`	Specifies the name of the cell (partition). The cell name is not required if the command is run on a top-level cell. *Data_type*: string, optional
`-corner corner_number`	Specifies the corner of the partition block. *Data_type*: int, optional
`-corner_to_pin_distance`	
	Specifies that the corner to pin distance constraints should be removed. *Data_type*: bool, optional
`-depth`	Specifies the pin depth related constraints should be removed. *Data_type*: bool, optional
`-edge`	Specifies that edge-related constraints should be removed. *Data_type*: bool, optional
`-global`	Specifies to reset global constraints. *Data_type*: bool, optional
`-help`	Outputs a brief description that includes the type and default information for each `reset_pin_constraint` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man reset_pin_constraint`
`-layer`	Specifies that the layer-related constraints should be removed. *Data_type*: bool, optional
`-location`	Specifies that the location-related constraints should be removed. *Data_type*: bool, optional

`-pins` *pin_name_list*	Specifies the name(s) of the partition pin(s). You can use wildcards (`*?`) with this parameter. *Data_type*: string, optional
`-pin_template { layer_id_or_name_list}`	
	Specifies the metal layers for the width, depth, spacing constraints at global level and cell level. *Data_type*: string, optional
`-side` *side_name*	Specifies name of the side. Valid values are { N S E W } or { L B T R } or all. *Data_type*: string, optional
`-spacing`	Specifies that the spacing-related constraints should be removed. *Data_type*: bool, optional
`-special_wire`	Specifies that the depth constraint that is honored for special wire pins by the `commit_partitions` command should be removed. *Data_type*: bool, optional
`-width` *width*	Specifies that the pin width related constraints should be removed. *Data_type*: bool, optional

Example

The following command removes the spacing- and layer-related constraints for all partition pins in the partition `sheet14`.

```
reset_pin_constraint sheet14 -pins * -layer -spacing
```

Related Information

- `set_pin_constraint`
- `get_pin_constraint`
- Partitioning the Design chapter in the *Innovus Stylus Common UI User Guide*
  - Setting Pin Constraints

route_hierarchical_clock

```
route_hierarchical_clock
[-help]
-dir ptn_parent_dir_name
-partition_net_dir ptn_net_dir_name
[-partition ptn_net_dir_name]
```

Runs `route_global_detail` on the nets specified in the net file (`<ptn_name>.net` file) provided for each of the specified partitions, in parallel, using EDP (parallel/distributed) framework.

Note:

Before running the `route_hierarchical_clock` command if you have a requirement to set any custom preferences or callbacks (For example,VIA preferences, routing layers preference etc), or perform any other operation, a file with name *`hier_clock_route_global_detail_route.tcl`* must be created in current run directory.

The *`hier_clock_route_global_detail_route.tcl`* file must have a procedure defined like below:

```
proc ::hier_clock_route_pre_global_detail_route {ptnName} {
Puts "called ::hier_clock_route_pre_global_detail_route $ptnName"
<body: operations/settings to be written in tcl format>
. . . .
}

proc ::hier_clock_route_post_global_detail_route {ptnName} {
Puts "called ::hier_clock_route_post_global_detail_route $ptnName"
<body: operations/settings to be written in tcl format>
. . . .
}
```

For example,

```
proc ::hier_clock_route_pre_global_detail_route {ptnName} {
Puts "called ::hier_clock_route_pre_global_detail_route $ptnName"
set bot_layer 6
set top_layer 9
set_db design_bottom_routing_layer $bot_layer
set_db design_top_routing_layer $top_layer
reset_db add_route_vias_auto
reset_db add_route_vias_ndr_only
set_db add_route_vias_ndr_only true

}

proc ::hier_clock_route_post_global_detail_route {ptnName} {
Puts "called ::hier_clock_route_post_global_detail_route $ptnName"
reset_db add_route_vias_auto
reset_db add_route_vias_ndr_only

}
```

Parameters

`-dir ptn_parent_dir_name`	Specifies the directory in which partition databases are present. *Data_type*: string, required
`-help`	Outputs a brief description that includes the type and default information for each `route_hierarchical_clock` parameter. For a detailed description of the command and all of its parameters, use the man command: `man route_hierarchical_clock`
`-partition_net_dir ptn_net_dir_name`	Specifies the directory in which net file for each partition is present. The net file contains the nets to be routed. *Data_type*: string, required
`-partition ptn_net_dir_name`	Specifies the list of partition names. *Data_type*: string, optional

Example

The following command routes the nets specified in the net files (./nets/tdsp_core.txt and ./nets/controller.txt), within the ./nets partition net directory, provided for each of the specified partitions (tdsp_core and controller)

```
route_hierarchical_clock -dir ./ptns -partition_net_dir ./nets -partition {tdsp_core controller}
```

where,

- ./ptns directory contains partitions, tdsp_core and controller.
- ./nets/tdsp_core.txt contains net names: N1, N2, N3
- ./nets/controller.txt contains net names: N4, N5

The result of this command is equivalent to the following tcl code:

```
 foreach ptnName $ptnNameList {
       cd $ptnParentDir/$ptnName
       read_db . $ptnName
       # select nets specified in $ptnNetDir/$ptnName.net
       # set routeGuide, if found in $ptnNetDir/$ptnName.rgf
       # source user-defined-procs for global detail route
       if {[file exists ${current_run_dir}/hier_clock_route_global_detail_route.tcl]} {
            source ${current_run_dir}/hier_clock_route_global_detail_route.tcl
       }
       # invoke call back : pre global detail route
       if {[info procs ::hier_clock_route_pre_global_detail_route] != {}} {
           ::hier_clock_route_pre_global_detail_route $ptnName
       }

       # run global NanoRoute on the selected nets
       set_db route_design_selected_net_only true
       route_global_detail
       set_db route_design_selected_net_only false

       # invoke call back : post global detail route
       if {[info procs ::hier_clock_route_post_global_detail_route] != {}} {
           ::hier_clock_route_post_global_detail_route $ptnName
       }

       # write def file
       write_def -routing $ptnName.def.gz
       }
```

Related Information

- route_global_detail
- Partitioning the Design chapter in the *Innovus Stylus Common UI User Guide*

set_blackbox_master_orient

```
set_blackbox_master_orient
[-help]
black_box_cell
{r90 r180 r270 mx mx90 my my90}
new_x_origin
new_y_origin
```

Converts the orientation of the master instance blackboxes from R0 to the specified orientation.

Parameters

black_box_cell	Specifies the name of the blackbox cell. *Data_type*: string, required
-help	Outputs a brief description that includes the type and default information for each `set_blackbox_master_orient` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man` set_blackbox_master_orient
new_x_origin	Specifies the new origin x-coordinate. *Data_type*: float, required
new_y_origin	Specifies the new origin y-coordinate. *Data_type*: float, required
{r90 r180 r270 mx mx90 my my90}	
	Specifies the new orientation value of the cell. The orientation value can be r90 r180 r270 mx mx90 my my90. *Data_type*: enum, required

Examples

- The following commands converts the orientation of the master blackboxes CORE_CELL120 to r90.
  `set_blackbox_master_orient CORE_CELL120` {r90}

Related Information

- `create_blackbox`
- `convert_blackbox_to_fence`
- `convert_fence_to_black_box`
- `delete_black_boxes`
- `set_blackbox_master_to_r0`
- Partitioning the Design chapter in the *Innovus Stylus Common UI User Guide*
  - Handling of Blackboxes with Non-R0 Orientation

set_blackbox_master_to_r0

```
set_blackbox_master_to_r0
[-help]
[-check_only]
[-cells cellName]
```

Converts the orientation of the master instance blackboxes to R0. Partitioning-related commands such as `assign_partition_pins`, `commit_partitions`, `assemble_design`, `flatten_partitions`, `convert_blackbox_to_fence`, and `edit_pin` work only with those blackboxes whose master instances have an R0 orientation. If the design has master blackboxes whose orientation is non-R0, you can use the `set_blackbox_master_to_r0` command to change the orientation of these master blackboxes to R0.

Note: Orientations of corresponding clone instances are also relatively transformed.

Parameters

`-check_only`	Specifies that the command will not actually convert the orientation of any master blackbox; it only displays the number of master blackboxes whose orientation would have been changed had the command been run without the `-check_only` parameter. *Default:* The orientation of master blackboxes in the design are converted to R0. *Data_type*: bool, optional
`-cells cellName`	Specifies that cell, or the list of blackbox cells, whose orientation will be converted to R0. *Default:* The orientation of all master blackboxes is converted to R0. *Data_type*: string, optional
`-help`	Outputs a brief description that includes the type and default information for each `set_blackbox_master_to_r0` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man set_blackbox_master_to_r0`

Examples

- The following commands converts the orientation of all master blackboxes that have a non-R0 orientation to R0.
  ```
  set_blackbox_master_to_r0
  ```
- The following commands converts the orientation of the master blackboxes CORE_CELL120 and CORE_CELL121 to R0.
  ```
  set_blackbox_master_to_r0 {CORE_CELL120 CORE_CELL121}
  ```
- The following command uses the `-check_only` parameter. The command, therefore, does not actually convert the orientation of any master blackboxes; it only returns the number of blackboxes whose orientation would have been converted to R0 had the command been run without the `-check_only` parameter.
  ```
  set_blackbox_master_to_r0 -check_only
  ```

Related Information

- `create_blackbox`
- `convert_blackbox_to_fence`
- `convert_fence_to_black_box`
- `delete_black_boxes`
- `set_blackbox_master_orient`
- Partitioning the Design chapter in the *Innovus Stylus Common UI User Guide*
  - Handling of Blackboxes with Non-R0 Orientation

set_clone_partition_orient

```
set_clone_partition_orient
[-help]
hinst
{r0 r90 r180 r270 mx mx90 my my90}
```

Changes the orientation of the specified partition clone.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `set_clone_partition_orient` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man set_clone_partition_orient`.
hinst	Specifies the name of the partition clone whose orientation has to be changed. *Data_type*: (hinst), required
`{r0 r90 r180 r270 mx mx90 my my90}`	Specifies the orientation. The orientation can be one of the following: `r0`, `r90`, `r180`, `r270`, `mx`, `mx90`, `my`, and `my90`. *Data_type*: enum, required

Example

The following command sets the orientation of the partition clone name `TDSP_ARB_C1 to r270:`

```
set_clone_partition_orient TDSP_ARB_C1 r270
```

Related Information

- `get_clone_partition_orient`
- Partitioning the Design chapter in the *Innovus Stylus Common UI User Guide*
- Changing Partition Clone Orientation

set_partition_area_io

```
set_partition_area_io
[-help]
[-add_buffer bufferName | -no_add_buffer]
[-top extension]
[-bottom extension]
[-left extension]
[-right extension]
[-selected]
[-pin_on_boundary]
```

Specifies the name of the buffer to be used for feedthrough insertion in a flip chip design. Use this command before partitioning the flip chip design.

Parameters

-add_buffer *bufferName*	
	Specifies either the name of the buffer, or that there are no buffers to be inserted. *Default*: `-no_add_buffer` *Data_type*: string, optional
`-bottom` *extension*	Specifies how far, in microns, to expand the Area I/O cell box in the bottom direction when determining the size of the placement and routing obstructions to be pushed down. *Data_type*: int, optional
`-help`	Outputs a brief description that includes the type and default information for each `set_partition_area_io` parameter.For a detailed description of the command and all of its parameters, use the `man` command: `man set_partition_area_io`
`-left` *extension*	Specifies how far, in microns, to expand the Area I/O cell box in the left direction when determining the size of the placement and routing obstructions to be pushed down. *Data_type*: int, optional
`-no_add_buffer`	Specifies that no buffers are to be inserted. *Data_type*: bool, optional
`-pin_on_boundary`	Specifies the internal pin created to be placed at the intersection of the hole punch (placement obstruction) and the space reserved for the partition. If it is not specified, the default is that the internal pin is placed directly on top of the area I/O cell. *Data_type*: bool, optional
`-right` *extension*	Specifies how far, in microns, to expand the Area I/O cell box in the right direction when determining the size of the placement and routing obstructions to be pushed down. *Data_type*: int, optional
`-selected`	Specifies hole punches for all currently selected cells that overlap partitions if they are Area I/O cells. If you do not specify this parameter, the command specifies hole punches for all Area I/O cells that overlap partitions. *Data_type*: bool, optional
`-top` *extension*	Specifies how far, in microns, to expand the Area I/O cell box in the top direction when determining the size of the placement and routing obstructions to be pushed down. *Data_type*: int, optional

Related Information

- Partitioning the Design chapter in the *Innovus Stylus Common UI User Guide*

set_partition_pin_status

```
set_partition_pin_status
[-help]
[-selected_pin_status {fixed | placed | cover}]
[-pins {pinName | pinNameList}]
[-status {fixed | placed | unplaced | cover}]
{[-pins {pinName | pinNameList}
  -status {fixed | placed | unplaced | cover}
 [-partition partitionName]] }
```

Sets the pin status for a pin of a partition as fixed, placed, cover, or unplaced. The `set_partition_pin_status` command operates, like a lot of other pin commands, on the top cell in addition to on partitions. If the top cell name is specified, or if `set_partition_pin_status` receives a "*" as its first argument, the IO pin status is modified. You can use this command any time after assigning pins.

Note:

- Use this command to unassign blackbox pins before the blackbox is committed. To unassign blackbox pins after the blackbox is committed, use the `flatten_partitions` command instead.
- If this command is run on a blackbox that has one or more pins with a single LEF port but with multiple shapes within the LEF port, the command will remove these multiple shapes and create a single shape. If this command is run on a blackbox that has one or more pins with multiple LEF ports, the first LEF port will be processed as per the command and the remaining LEF ports will remain unchanged.

Parameters

`-partition` *partitionName*	Specifies the name of the partition. You can use wildcards (*?) with this parameter. *Data_type*: string, optional
`-help`	Outputs a brief description that includes type and default information for each `set_partition_pin_status` parameter. For a detailed description of the command and all of its parameters, use the man command: `man set_partition_pin_status`.
`-pins` *pinName* | *pinNameList*	
	Specifies the name of the pin(s) of the partition. You can use wildcards (*?) with this parameter. *Data_type*: string, optional
`-selected_pin_status {fixed | placed | cover}`	
	Selects pins with only specified status. This options enables the `set_partition_pin_status` command to change the status of selected pins based on their present status and not just the names. • `fixed`: Selects pins with fixed status. • `placed`: Selects pins with placed status. • `cover`: Selects pins with cover status. *Data_type*: enum, optional
`-status {fixed | placed | unplaced | cover}`	
	Specifies the status of the pin. • `fixed`: Fixed pin status specifies the pin's location, orientation, and that it's location cannot be moved by automatic tools, but can be moved by interactive commands. **Note**: Fixed pins can be moved by automatic pin assignment commands by using the `-moveFixed` parameter. • `placed`: Placed pin status specifies the pin's location, orientation, and that it's location is fixed, but can be moved during automatic layout. • `unplaced`: Unplaced pin status specifies that the pin does not have a location. • `cover`: Cover pin status specifies the pin's location, orientation, and that it is a part of the cover macro. A cover pin cannot be moved by automatic tools or by interactive commands. *Data_type*: enum, optional

Example

The following command marks all partition pins to fixed:

```
set_partition_pin_status -partition * -pins * -status fixed
```

Related Information

- `get_partition_pin_status`
- `flatten_partitions`
- `assign_partition_pins`
- Partitioning the Design chapter in the *Innovus Stylus Common UI User Guide*

set_partition_pin_use

```
set_partition_pin_use
[-help]
{design | partition}
pin
pin_use
```

Allows you to change the `USE` property of a partition pin.

Note: A partition pin can also be an I/O pin if the partition is a top-level cell.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `set_partition_pin_use` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man setPtnPinUse`
{design | partition}	Specifies the name of the partition, blackbox or top-level cell name. *Data_type*: string, required
pin	Specifies the pin name. *Data_type*: string, required
pin_use	The choices represent `USE` properties, SIGNAL | CLOCK | ANALOG, that you can change with this command. *Data_type*: string, required

Example

- The following command sets the `USE` property of pin `A` in partition `PTN1` to `CLOCK`:
  ```
  set_partition_pin_use PTN1 A CLOCK
  ```
- The following command sets the USE property for pin scan_clk to CLOCK:

  Original DEF:
  ```
  - scan_clk + NET scan_clk + DIRECTION INPUT + USE SIGNAL ;
  ```

  Run set_partition_pin_use to change USE to CLOCK:
  ```
  set_partition_pin_use [dbGet top.name] scan_clk CLOCK
  ```

  DEF after change:
  ```
  - scan_clk + NET scan_clk + DIRECTION INPUT + USE CLOCK ;
  ```

Related Information

- Partitioning the Design chapter in the *Innovus Stylus Common UI User Guide*

set_pin_constraint

```
set_pin_constraint
[-help]
{[-global | -cell cell_name | -match_route_early_global ]
[-global | -pins pin_name_list | -area {x1 y1 x2 y2} | -side side_names | -match_route_early_global ]
[-global | -layer {layer_id_or_name_list} | -location {x y z}]
[-global | -location {x y z} | -edge edge_numbers | -corner corner_number]
[[-pins pin_name_list [-location {x y z}] [-edge edge_numbers]] |
[ [-location {x y z}] [-edge edge_numbers]] |
[-corner_to_pin_distance pin_distance [-corner corner_number]]]
[{-pins pin_name_list | -pin_template {layer_id_or_name_list}}
[-use_min_width_as_depth | -depth depth] [-width width]]
[[-layer {layer_id_or_name_list} [-side side_names]] |
[-pin_template {layer_id_or_name_list} [-use_min_width_as_depth ]]]
[-spacing spacing [-area {x1 y1 x2 y2}] [-side side_names]]}
[-depth depth [-special_wire ]]
```

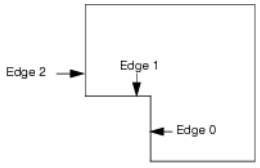
Sets the constraints for partition pins or an I/O pins. You can use this command before assigning pins. The constraints can be applied on three different levels:

- Global pin constraint
- Cell specific pin constraint
- Pin specific constraint

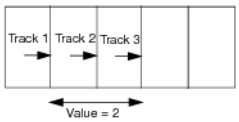
If no pin specific constraints exist, then the cell level constraints are applied. If cell specific pin constraints are not defined, then global constraints are applied.

Note: To retrieve the pin constraints specified using the `set_pin_constraint` command, use the `get_pin_constraint` command. To remove the constraints for a partition pin, use the `reset_pin_constraint` command.

Parameters

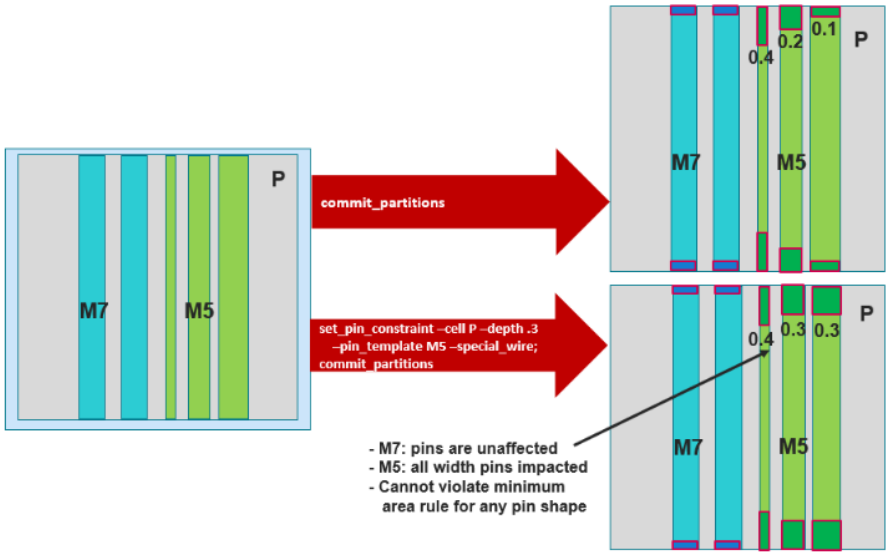
`-area {x1 y1 x2 y2}`	Specifies an area overlapping with partition where spacing constraint. *Data_type*: rect, optional
`-cell cell_name`	Specifies the name of the module. The module name is not required if the command is run on a top-level module. You can use wildcards (*?) with this parameter. *Data_type*: string, optional
`-corner corner_number`	
	Specifies the corner of the partition block. *Data_type*: int, optional
`-corner_to_pin_distance pin_distance`	
	Specifies the corner to pin distance in tracks. *Data_type*: int, optional
`-depth depth`	Specifies the pin depth in microns. *Data_type*: float, optional
`-edge edge_numbers`	Specifies the integer value where edge numbering starts from lower-left corner of a partition clock-wise. Edge `0` is the edge that has the smallest y value.  You can set multiple sides in the `-edge` option. This allows you to set multiple sides in a rectilinear design. *Data_type*: string, optional

`-global`	Specifies that the global constraints for spacing, width, and depth to be honored. **Note:** If no pin specific constraints exist, then the cell level constraints are applied. If cell specific pin constraints are not defined, then global constraints are applied. For example, the following command specifies that the global constraints are to be honored. `set_pin_constraint -global -spacing 2 -width 2 -pin_template { 2 3 4 }` *Data_type*: bool, optional
`-help`	Outputs a brief description that includes the type and default information for each `set_pin_constraint` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man set_pin_constraint`
`-layer {`*layerId* | *layerIdList*`}`	
	Specifies the metal layers for the partition pin. If you specify multiple layers, separate the layer names with a space. In this case, the command will constrain the pin to one of the specified layers. **Notes:** • This parameter does not restrict the pin to a location on the layer. If you want to restrict a pin on a particular location on a particular layer, use the `-location` parameter instead. • The `-layer` parameter cannot be used with the `-location` parameter. • The `-layer` parameter cannot be used with the `-spacing` parameter. • Layers can be specified using the LEF layer names or layer ID numbers. *Data_type*: string, optional
`-location {`*x y z*`}`	Specifies the x and y locations and z layer ID number of the partition pin. The x and locations are calculated with respect to the distance in microns from the lower left corner of the module. The layer number specified should be a preferred layer. Otherwise, the command will move the pin to a preferred layer. For example, `-location {10 12 7}` specifies that the pin should be constrained to a location 10 microns on the x axis and 12 microns on the y axis from the lower left corner of the partition on layer `Metal7`. **Notes:** • If the pin location as per the specified x and y locations is not on a track, the pin is snapped to the nearest track on the specified layer. • If you specify this parameter, the assigned pins have a *Placed* status. • The `-location` parameter cannot be used with the `-layer` parameter. *Data_type*: string, optional
`-match_route_early_global`	When specified, it changes the pin constraints to make them compliant with `route_early_global` and does the following for every partition: • Allows all layers from `design_top_routing_layer` to `design_bottom_routing_layer`. • Sets pin spacing as 1 **Note:** The changes in pin constraints impact the output of the pin assignment commands - `assign_partition_pins`, `assign_io_pins`, and `edit_pin`. *Data_type*: bool, optional
`-pins` *pin_name_list*	Specifies the name(s) of the partition pin(s). You can use wildcards (`*?`) with this parameter. *Data_type*: string, optional
`-pin_template {`*layer_id_or_name_list*`}`	
	Specifies the metal layers for the width, depth and spacing constraints at global level and cell level *Data_type*: string, optional
`-side` *side_name*	Specifies name of the side. Valid values are { N S E W } or { L B T R } or all. *Data_type*: string, optional

`-spacing min_space`	Specifies the minimum spacing between pins in pin pitch value, that is, the distance between tracks. For example, if you specify a value of `2`, the minimum pin spacing will be equal to the distance between track 1 and track 3.  *Data_type*: int, optional
`-special_wire`	Specifies the depth constraint to be set and honored for special wire pins. When `-special_wire` parameter is specified with the `-depth`, `-cell` and `-pin_template` parameters, it enables the depth on desired layers of specified cells to be honored for special wire pins by the `commit_partitions` command. *Data_type*: bool, optional
`-use_min_width_as_depth`	Minimum width should be used as the pin depth. *Data_type*: bool, optional
`-width width`	Specifies the pin width in microns. *Data_type*: float, optional

Example

- The following command specifies a width of 10 micrometers for all partition pins in the partition `sheet14`. These pins will be placed either on layer 2 or on layer 4.
  ```
  set_pin_constraint -cell sheet14 -pins * -layer {Metal2 Metal4}
  ```
- The following commands set the pin width and depth of layer `Metal2` for partition cell `ALU` to 0.4 and 0.6 respectively.
  ```
  set_pin_constraint -cell ALU -layer Metal2 -width 0.4
  set_pin_constraint -cell ALU -layer Metal2 -depth 0.6
  ```
- The following commands set the pin width of pin pGroup1 to 0.3 and pin depth of pin pGroup1 to default.
  ```
  set_pin_constraint -cell ALU -pins pGroup1 -width 0.3
  set_pin_constraint -cell ALU -pins pGroup1 -depth default
  ```
  With this example, all the pins of pGroup1 will have the width 0.3 and the default depth.
- The following commands set the minimum pin spacing for top and bottom edge of partition cell ALU to 1 track.
  ```
  set_pin_constraint -cell ALU -edge T -spacing 1
  set_pin_constraint -cell ALU -edge B -spacing 1
  ```
- The following command sets minimum pin spacing for all edges of partition cell TDSP to 3 tracks
  ```
  set_pin_constraint -cell TDSP -all -spacing 3
  ```
- The following command sets the sides, T and B, of layer 3 and 5 of partition cell ptn4
  ```
  set_pin_constraint -cell ptn4 -layer {3 5} -side {T B}
  ```
- The following command sets the edges, 2 and 0, of partition cell ptn4
  ```
  set_pin_constraint -cell ptn4 -pins * -edge {2 0}
  ```
- The following commands sets the pin depth of layer `M5` for partition cell `P` to 0.3 using the `-special_wire` parameter so that it is honored only by the `commit_partitions` command.
  ```
  set_pin_constraint-cell P -depth .3 -pin_template M5 -special_wire;
  commit_partitions
  ```



Related Information

- `get_pin_constraint`
- `reset_pin_constraint`
- Partitioning the Design chapter in the *Innovus Stylus Common UI User Guide*
  - Setting Pin Constraints

set_promoted_macro_pin

```
set_promoted_macro_pin
[-help]
{-reset  | -report_only_file filename.tcl |
 {[-insts {inst_name | inst_name_list}]
  [-pins {pin_name | pin_name_list}]
  [-layers {layer_id | layer_id_list}]
  [-override ]
  [-abutted_to_boundary_only]}}
```

Selects or marks signal and/or PG pin shapes to be promoted later on. You can use this command before assigning pins.

Note: The `set_promoted_macro_pin` command should be called after specifying the maximum and minimum metal layer names for promoting macro pins using the `assign_pins_promoted_macro_max_layer` and `assign_pins_promoted_macro_min_layer` attributes, respectively.

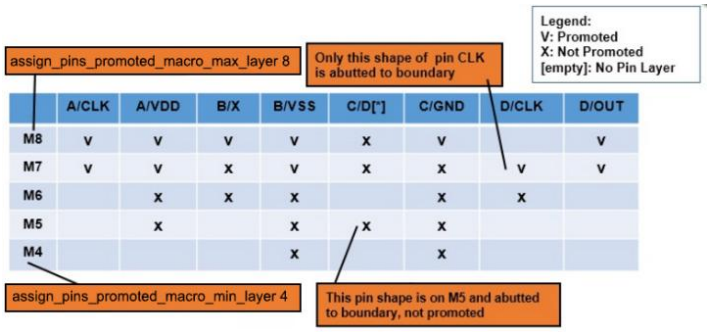
Parameters

`-abutted_to_boundary_only`	Specifies that all macro pin shapes that are abutted to the partition boundary will be selected for promotion. The `-pins` parameter can also be used to select abutted pins. **Note:** You can use the `assign_pins_max_channel_width_as_abutted` attribute to enable thin channels in the design to be considered as abutted partitions in the pin assignment flow. *Data_type*: bool, optional
`-help`	Outputs a brief description that includes the type and default information for each `set_promoted_macro_pin` parameter. For a detailed description of the command and all of its parameters, use the man command: `man set_promoted_macro_pin`
`-insts {inst_name | inst_name_list}`	
	Specifies the name of instances. You can also use wildcard (* or ?) for specifying the instance names. This parameter is optional. If `-insts` is not specified, all BLOCK, PAD, and AREAIO macros are selected. This is similar to specifying `-insts *` *Data_type*: string, optional
`-layers {layer_id | layer_id_list}`	
	Specifies the metal layer(s) of the macro pins. This parameter is optional. If `-layers` is not specified, all pin layers specified between the `assign_pins_promoted_macro_max_layer` and the `assign_pins_promoted_macro_min_layer` attributes range are selected. *Data_type*: string, optional
`-override`	Specifies to override specific option(s) where previous settings of these options will be removed and be replaced by new specified value(s). **Note:** All `set_promoted_macro_pin` commands are incremental with no conflict resolution, unless the `-override` option is used. *Data_type*: bool, optional
`-pins {pin_name | pin_name_list}`	Specifies the name of macro pins. You can also use wildcard (* or ?) for specifying the pin names. This parameter is optional. If `-insts` is not specified, all signal and P/G pins are selected. This is similar to specifying `-pins *` *Data_type*: string, optional
`-report_only_file filename.tcl`	Writes out all `set_promoted_macro_pin` settings in memory into an output TCL file. *Data_type*: string, optional
`-reset`	Resets all or specified promoted pin selections. *Data_type*: bool, optional

Examples

- The following command promotes the top most layer pins of selected macros

```
set_db assign_pins_promoted_macro_min_layer 4
set_db assign_pins_promoted_macro_max_layer 8
set_promoted_macro_pin -reset
set_promoted_macro_pin -insts {A B} -pins * -layers {7 8}
set_promoted_macro_pin -insts B -pins X -layers 8 -override
set_promoted_macro_pin -insts C -layers 5 -abutted_to_boundary_only
set_promoted_macro_pin -insts C -layers {8}
set_promoted_macro_pin -insts C -pins D[*] -layers {} -override
set_promoted_macro_pin -insts D -pins OUT
set_promoted_macro_pin -insts D -pins CLK -abutted_to_boundary_only
assign_io_pins
```



	A/CLK	A/VDD	B/X	B/VSS	C/D[*]	C/GND	D/CLK	D/OUT
M8	v	v	v	v	x	v		v
M7	v	v	x	v	x	x	v	v
M6		x	x	x		x	x	
M5		x		x	x	x		
M4				x		x		

Related Information

- `assign_io_pins`
- `assign_pins` Category Attributes
- Partitioning the Design chapter in the *Innovus Stylus Common UI User Guide*

undo_uniquify_partition

```
undo_uniquify_partition
[-help]
-hinst hierarchical_instance_name
[-preserve_routes]
```

Brings back the recursion of cells in a netlist and restores the master and clone association.

Note: The `uniquify_partition` command is used to remove the master and clone association by uniquifying the netlist.

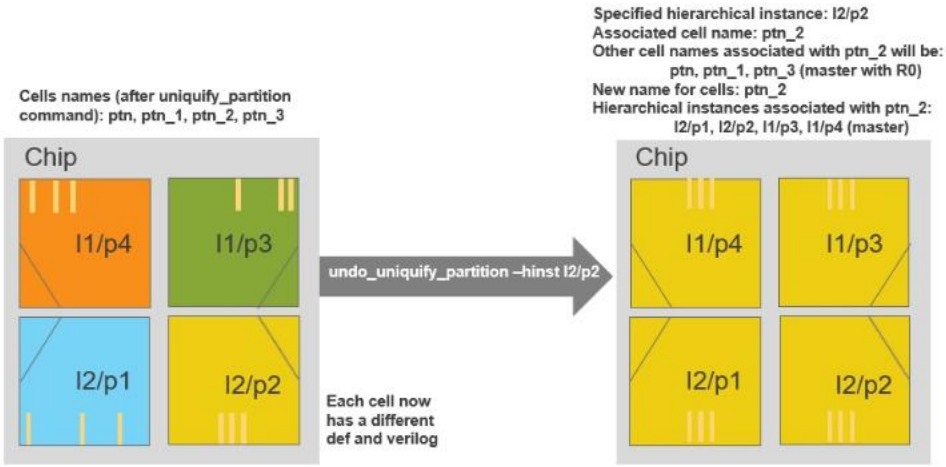
Parameters

`-help`	Outputs a brief description that includes the type and default information for each `undo_uniquify_partition` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man undo_uniquify_partition`
`-hinst hierarchical_instance_name`	Specifies name of the hierarchical instance that is used to bring back the recursion of cells in a netlist and restore the master and clone association. Based of the selected hierarchical instance, its cell (module) is identified. All other occurrences of the identified cell with the same naming format (`cellname_<number>`) are then renamed to the identified cell name. That means all instances will have the same cell name and definition again, thereby restoring the master and clone relationship. **Note:** • You can specify any hierarchical instance with the `undo_uniquify_partition` command. • Any hierarchical instance with R0 orientation can become a master. • The name of the cells will change to the name of the cell of the specified hierarchical instance. • The netlist and def of the cell of the hierarchical instance name specified with the `undo_uniquify_partition` command will be copied in all hierarchical instances. *Data_type*: string, optional
`-preserve_routes`	Preserves the routes during `undo_uniquify_partition`. *Data_type*: bool, optional

Example

The following command brings back the master and clone association for the cell name associated with `I2/p2`:

```
undo_uniquify_partition -hinst I2/p2
```



Related Information

- uniquify_partition
- Partitioning the Design chapter in the *Innovus Stylus Common UI User Guide*

uniquify_partition

```
uniquify_partition
[-help]
-hinst hierarchical_instance_name
```

Removes the master and clone association by uniquifying the cells (module) in the netlist.

Note: You can use the `undo_uniquify_partition` command to remove the uniquification and bring back the master and clone association.

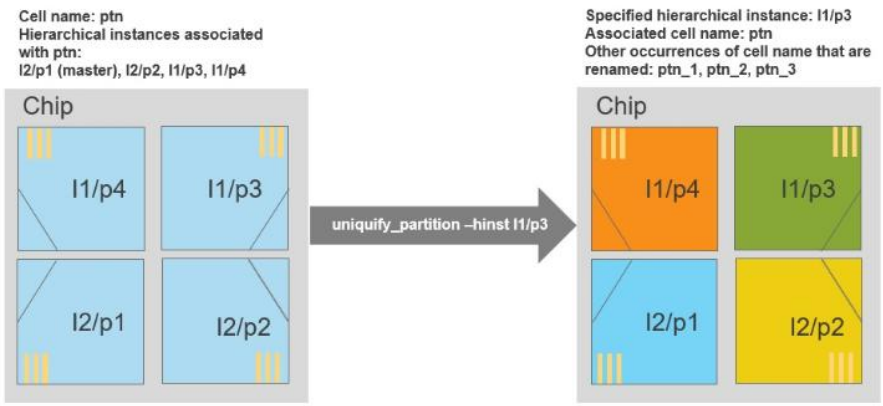
Parameters

`-help`	Outputs a brief description that includes the type and default information for each `uniquify_partition` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man uniquify_partition`
`-hinst hierarchical_instance_name`	Specifies the name of the hierarchical instance to uniquify. Based of the specified hierarchical instance, its cell (module) is identified. All other occurrences of the identified cell are renamed by adding the `_<number>` suffix (for example `cellname_<number>`) in order to uniquify the cell. **Note:** • You can specify any hierarchical instance with the `uniquify_partition` command. It does not necessarily have to be a master. • The `uniquify_partition` command preserves the orientation of the hierarchical instances. • The `uniquify_partition` command preserves the pin location of the hierarchical instances. • A new cell name is created for each hierarchical instance. *Data_type*: string, required

Example

The following command identifies the cell associated with `I1/p3` and then renames all other occurrences of the cell in order to remove the master and clone association.

```
uniquify_partition -hinst I1/p3
```



Related Information

- `undo_uniquify_partition`
- Partitioning the Design chapter in the *Innovus Stylus Common UI User Guide*

update_net_group

```
update_net_group
[-help]
-name
-add_nets {netName | netNameList}
```

Adds a net to specified net group that was created with the `create_net_group` command. A net can be a single net, bus name, or segment of a bus. You can use this command after defining a net group.

Note: If a net is a part of a net group, the net cannot belong to any other net group.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `update_net_group` parameter. For a detailed description of the command and all of its parameters, use the man command: `man update_net_group`
`-name`	Specifies the name of the created net group. *Data_type*: string, required
`-add_nets {`*`netName	netNameList`*`}`	
	Specifies the name of an individual net, bus, or segment of a bus. You can use wildcards (*?) with this parameter. *Data_type*: string, required

Examples

- The following command adds net ENOUT to net group bundle1:
  ```
  update_net_group bundle1 -add_nets ENOUT
  ```
- The following command adds the entire bus cpua to net group bundle2:
  ```
  update_net_group bundle2 -add_nets cpua
  ```
- The following command adds only bus segments 2 through 10 to net group bundle1:
  ```
  update_net_group bundle1 -add_nets cpua[2:10]
  ```

Related Information

- `create_net_group`
- `delete_net_group`
- `delete_nets_from_net_group`
- Partitioning the Design chapter in the *Innovus Stylus Common UI User Guide*
  - Setting Pin Constraints
  - Net Group

update_partition_cell_obs

```
update_partition_cell_obs
[-help]
[partition_name]
[-no_pin_cut_space | -has_pin_cut_space]
[-no_via_cut_space | -has_via_cut_space]
```

Recreates cell blockage for a specified partition, or all partitions. This reflects incremental changes, such as pin movement. You can use this command after editing the pin in a partition with the Pin Editor, or using the `edit_pin` command, to move pins in the partition.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `update_partition_cell_obs` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man update_partition_cell_obs`
partition_name	Specifies the partition to recreate the cell blockage. *Default*: All partitions *Data_type*: string, optional
`-no_pin_cut_space`	Specifies that no cut space will be created for pins in the partition cell blockage. **Note:** Blockages are always cut for power pins. *Data_type*: bool, optional
`-has_pin_cut_space`	Specifies that cut space will be created for pins in the partition cell blockage. **Note:** This is the default. *Data_type*: bool, optional
`-no_via_cut_space`	Specifies that no cut space will be created for vias in the partition cell blockage. *Data_type*: bool, optional
`-has_via_cut_space`	Specifies that cut space will be created for vias in the partition cell blockage. **Note:** This is the default. *Data_type*: bool, optional

Example

The following command recreates pin blockages for partition `EUG`:

```
update_partition_cell_obs EUG
```

Related Information

- `edit_pin`
- Partitioning the Design chapter in the *Innovus Stylus Common UI User Guide*

update_partition_cut

```
update_partition_cut
[-help]
[-partition string]
-cut_area x1 y1 x2 y2
```

Creates a partition rectangular cut from a partition. This object can be created after specifying a module guide to be a partition. Rectangular cuts can only occur on the edge of the partition guide. You can use this command after specifying partition.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `update_partition_cut` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man update_partition_cut`
`-cut_area x1 y1 x2 y2`	Specifies the coordinates of the cut area (*x1 y1 x2 y2*). *Data_type*: rect, required
`-partition string`	Specifies a partition to create a partition cut. If this option is not selected, the command loops through all the partitions an attempts to find the overlapped partition, then creates a partition cut for that partition. *Data_type*: string, optional

Example

The following command creates a partition cut area to modify a rectangular partition:

```
update_partition_cut 2353.8000 7123.000 2653.8000 8298.2000
```

Related Information

- Partitioning the Design chapter in the *Innovus Stylus Common UI User Guide*

update_pin_group

```
update_pin_group
[-help]
[-partition cellName]
-pin_group pinGroupName
-pins {pinName | pinNameList}
```

Adds a module port to specified pin group that was created by `create_pin_group` command. You can use this command after defining a net group.

Note: You can use this command for the top-level also.

Parameters

`-partition cellName`	Specifies the name of a module. This parameter is not required for top-level ports. *Data_type*: string, optional
`-help`	Outputs a brief description that includes the type and default information for each `update_pin_group` parameter. For a detailed description of the command and all of its parameters, use the man command: `man update_pin_group`
`-pin_group pinGroupName`	
	Specifies the name of the created pin group. *Data_type*: string, required
`-pins {pinName | pinNameList}`	
	Specifies the name of a module port. You can specify wildcards (*?) with this parameter. *Data_type*: string, required

Example

- The following command adds the module SH17's port RASN to pin group pin17:
  ```
  update_pin_group -partition SH17 -pin_group pin17 -pins RASN
  ```
- The following command adds the top-level port ADDER to pin group PIN_GRP_TOP
  ```
  update_pin_group -pin_group PIN_GRP_TOP -pins ADDER
  ```

Related Information

- `create_pin_group`
- `delete_pin_groups`
- Partitioning the Design chapter in the *Innovus Stylus Common UI User Guide*
  - Setting Pin Constraints
  - Net Group

write_black_box_lef

```
write_black_box_lef
[-help]
{-insts instName | -all}
[lef_file]
```

Creates a blackbox LEF file containing the pin definitions and obstructions.

Note: The `write_partitions` command also automatically saves this file, among others.

Parameters

`-all`	Specifies all blackbox instances. *Data_type*: bool, optional
`-help`	Outputs a brief description that includes the type and default information for each `write_black_box_lef` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man write_black_box_lef`
`-insts instName`	Specifies the name of the blackbox instance. You can use wildcards (`*?`) with this parameter. *Data_type*: string, optional
lef_file	Specifies the LEF filename. If you do not specify a name, the default is *cellTypeName*`.lef`. *Data_type*: string, optional

Example

The following command creates a LEF file containing the pin definitions and obstructions for all blackbox instances that start with the letter `A` to a LEF file named `BBox.lef`:

```
write_black_box_lef A* lef_file BBox.lef
```

Related Information

- `write_partitions`
- `create_blackbox`
- Partitioning the Design chapter in the *Innovus Stylus Common UI User Guide*

write_generalized_feedthru_paths

```
write_generalized_feedthru_paths
[-help]
[-dont_generalize_reverse_paths]
[-dont_generalize_sub_paths]
[-topological_out_file file_name]
-use_topological_file file_name
```

Automatically derives feedthrough topological paths from the user specified paths to cover all the chain connectivity probabilities in a master and clone hinst scenario. It reads an input topological file, in version 0.5 and creates the version 2.0 topological file.

Note: For the given set of master-clone hinst connectivity, based on floorplan, multiple user defined paths may be required for other same set hinst master and clones. The `write_generalized_feedthru_paths` command can automatically choose correct path for individual master and clone hinst, based on matching connectivity and floorplan, from the set of paths provided by the user.

Parameters

`-dont_generalize_reverse_paths`	
	When enabled, the generalization algorithm does not generalize reverse chain paths. *Data_type*: bool, optional
`-dont_generalize_sub_paths`	
	When enabled, the generalization algorithm does not generalize sub-paths. *Data_type*: bool, optional
`-help`	Outputs a brief description that includes the type and default information for each `write_generalized_feedthru_paths` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man write_generalized_feedthru_paths`
`-topological_out_file file_name`	
	Specifies the name of the output topo 2.0 file that contains all paths for all master and clone chain. **Note:** This file should be used as an input file for the `add_partition_feedthrus` command. *Default:* path.topo *Data_type*: string, optional
`-use_topological_file file_name`	
	Specifies the user specified feedthrough path guidance file. The input topological file with new version 0.5 format contains a path for one master and clone chain/tree to derive path of all master and clone chains. **Note:** You can use the *Path Guidance GUI* form (*Partition - Feedthough Ports - Path Guidance)* to generate the topological file with new version 0.5 format. *Data_type*: string, required

Example

- The following command reads the topology file “a.topo” in version 0.5 and creates the version 2.0 topological file “path.topo”:

  ```
  write_generalized_feedthru_paths  -use_topological_file a.topo
  ```

- The following command reads the topological file “a.topo” in version 0.5 and creates the version 2.0 topological file “new.topo”:

  ```
  write_generalized_feedthru_paths  -use_topological_file a.topo -topological_out_file new.topo
  ```

Related Information

- `add_partition_feedthrus`
- Partitioning the Design chapter in the *Innovus Stylus Common UI User Guide*

write_net_groups_topo

```
write_net_groups_topo
[-help]
```

Creates a topological file (used by the `add_partition_feedthrus` command) for nets of the net group associated with a bus guide. The topology of nets for feedthrough is derived from the shape of the bus guide. All the partitions that the bus guide shape overlaps with, become a node in the topological path.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `write_net_groups_topo` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man write_net_groups_topo`

Example

- The following command writes the net group information to a topology file.
  ```
  write_net_groups_topo
  ```

Related Information

- Partitioning the Design chapter in the *Innovus Stylus Common UI User Guide*

write_partition_pins

```
write_partition_pins
[-help]
{-partition partition_name | -all | -design| -selected}
out_file
```

Saves the pin assignment information for one or more partitions. You can subsequently load the saved pin assignment information with the `read_partition_pins` command. The `write_partition_pins` command also saves the pin assignment information for blackboxes.

Parameters

`-all`	Specifies that pin assignment information will be saved for all partitions, including the top-level design. *Default*: By default, the `write_partition_pins` command saves the pin assignment information for all partitions. *Data_type*: bool, optional
`-design`	Specifies that pin assignment information will be saved only for the top-level design. *Data_type*: bool, optional
`out_file`	Specifies the name of the file in which the partition pin assignment information will be saved. *Data_type*: string, required
`-help`	Outputs a brief description that includes the type and default information for each `write_partition_pins` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man write_partition_pins`
`-partition` *`partition_name`*	Specifies the name of the partition for which the pin assignment information is to be saved. *Data_type*: string, optional
`-selected`	Specifies that the pin assignment information will be saved for all the selected pins of the partitions, including the top-level design. *Data_type*: bool, optional

Example

The following command saves the pin assignment information for the partition named `tdsp_block` into a file named `tdsp_block_pin_info`.

```
write_partition_pins -partition tdsp_block tdsp_block_pin_info
```

Related Information

- `assign_partition_pins`
- `read_partition_pins`
- `reset_partition_pins`
- Partitioning the Design chapter in the *Innovus Stylus Common UI User Guide*
  - Saving the Partition Pins

write_partitions

```
write_partitions
[-help]
[partitions {list}]
[-floorplan_only]
[-no_std_cells]
[-no_trial_route]
[-verilog]
[-dir dirName]
[-module_model_tag tag_name]
[{[-no_floorplan | {[-no_netlist] [-def_unplaced]}]
[-def [-def_unplaced]]
[-def_no_cut_row]
[-top_level_no_partition_module]
{{[-partition_lib ptnLibName] [-partition_view ptnViewName] [-top_view topViewName]}}]]
```

Saves the partition information to the current directory or a specified directory. This command is run after running the `commit_partitions` command to generate timing budget files and stamp model files.

Notes:

- The `write_partitions` command does not write GNC rules in the partition floorplan file for multiple-supply voltage (MSV) designs.The Common Power Format (CPF) file contains this information.
- The `write_partitions` command supports partitions with flexmodels. If there are flexmodels at the top-level, then the saved top-level netlist contains flexmodels. If there are no flexmodels outside partitions, then the top-level netlist will be the original full netlist. Similarly, if a partition has flexmodels, then its partition netlist will contain flexmodes.
- The `write_partitions` command saves the design information to the global init_* variables in the *.global file.
- The `write_partitions` command supports multi-threading. This reduces the run time for huge designs by saving individual blocks in parallel.
- The `write_partitions` command saves mmmc data without timing information for partitions for which timing budgets are not derived.
- The `write_partitions` command automatically saves all the bus guide constraints with associated net groups. This enables bus routing at block level as per chip level constraints.
- The `write_partitions` command automatically saves all the cell padding information to each partition block data. This information is saved in partitioned block .fp file.
- The `write_partitions` command inherits all the attributes assigned to a net for block implementation flow. Preserving these attributes allows the Early Global Router and the NanoRoute Router to route the nets following specific requirements.
- The `write_partitions` command writes out a DEF file for each block with its scan information in the `<partition_name>.scandef.gz` format.
- The `write_partitions` command saves an additional file in each parent directory in the `<design_name>.pg.gz` format to support for PG-file in hierarchical flow. The `assemble_design` command loads this file when it is run with the `-fe` mode.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `write_partitions` parameter. For a detailed description of the command and all of its parameters, use the man command: `man write_partitions`
`-def`	Writes out the DEF format files for the design. *Data_type*: bool, optional
`-def_no_cut_row`	Specifies that, when row information is written to the DEF file, the rows are not cut against placement obstructions, block halos, or power domain fence minimum gaps. *Data_type*: bool, optional
`-dir dirName`	The directory name that will contain the partition information/files. *Data_type*: string, optional

`-floorplan_only`	Writes out incremental floorplan by pushing down floorplan changes from a full chip level design to a partition block. This option allows you to run `write_partitions` for a uncommitted partition without running the `partition` command. For committed partitions, `write_partitions -floorplan_only` reports an error. The `write_partitions -floorplan_only -dir` command honors blackboxes and saves limited floorplan information for blackboxes in the previously saved data created by the `write_partitions -dir` command. **Note:** After `write_partitions -floorplan_only` command, a non-committed partition still remains uncommitted. *Data_type*: bool, optional
`-def_unplaced`	When used with the `-def` parameter, specifies that the unplaced standard cells should also be saved to the DEF file. This is equivalent to the `-unplaced` parameter of the `write_def` command. When used with the `-pdef` parameter, specifies that the placed standard cells should also be saved to the PDEF file. *Data_type*: bool, optional
`-module_model_tag` *tag_name*	Specifies the module model tag to be used for saving the partition. This creates pnr and lef module models for each partition block. For top-level design, this also creates a pnr module model. This parameter is part of the iHDB flow. Before using this parameter, you must set the default central directory name (`set_module_model -default_dir` *default_model_data_directory*) where the module models data will be saved/loaded. **Note:** These block-level and top-level pnr models should be used for implementing the block-level and top-level designs, respectively. For example, if design dtmf has the ptn_wrapper partition and the default global data directory is /myproject/DATA. With `write_partitions -module_model_tag init,` a pnr model will be created for top-level design in the following directory: `myproject/DATA/dtmf/init/pnr`. For partition ptn_wrapper, a pnr and lef models will be created in the `/myproject/DATA/ptn_wrapper/init/pnr` and `/myproject/DATA/ptn_wrapper/init/lef` directories. *Data_type*: string, optional
`-no_floorplan`	Saves partitions without generating a floorplan file for the partitioned design. *Data_type*: bool, optional
`-no_netlist`	Saves partitions without generating a netlist file for the partitioned design. *Data_type*: bool, optional
`-no_std_cells`	When you specify this option, standard cells are not written in the DEF file. *Data_type*: bool, optional
`-no_trial_route`	When you specify this option, trial routed wires are not written in the DEF file. *Data_type*: bool, optional
partitions {*list*}	Specifies the names of each partition to be saved. *Data_type*: string, optional
`-partition_lib` *ptnLibName*	Specifies an OpenAccess database library directory where the top-level design and the block-level design for all the partitions will be saved. This parameter is used when saving the partition into an OpenAccess database. **Note:** For OpenAccess data, you can save the specified partitions using the *partitions* parameter. However, if the specified partition is a top-level design then all committed partitions must be saved just like no partition names are specified to avoid unbound references in OpenAccess. *Data_type*: string, optional
`-partition_view` *ptnViewName*	

	Specifies the view name for the top and the partition views. This parameter is used when saving the partition into an OpenAccess database. *Default:* The default value of the view is layout **Note:** • If `-partition_view` is specified, but not `-partition_lib`, then the current design lib is used. You do not have to use the `write_partitions` command to create a special library to hold the partition data. You can create the partition data into the same design library that holds the full-flat version of the design. In this case, cell views will be overwritten if they already exist. • If `-partition_view` is specified, but not `-top_view`, then all cell views created with the `write_partitions` command will have the *`ptnViewName`* appended to the top cell's cellview. In order to prevent overwriting the top cell's cellview if the name already exists, use the `-top_view` option to specify a different view name than the one used for the rest of the partitions being created by the `write_partitions` command. • If `-dir` is specified, but `-partition_view` is not, then the `write_partitions` command would be in the non-OA mode. If `-dir` and `-partition_view` are specified then the `dir` is created, but instead of putting design data into it, only the `cds.lib` is added to the directories created. *Data_type*: string, optional
`-top_level_no_partition_module`	Suppresses stub verilog modules for partitions in the top-level partition netlist. *Data_type*: bool, optional
`-top_view` *`topViewName`*	Specifies view name for top design. *Data_type*: string, optional
`-verilog`	Writes out an ASCII Verilog file to the saved partition directory. The netlist is always stored as a binary file, so this Verilog file is not used by `read_db`. It is sometimes useful to save the Verilog in addition to the other saved files to make it possible to send the Verilog directly to other tools, or to open it in an editor for debugging. *Data_type*: bool, optional

Example

- The following command saves the partition information/files in a directory `try3` hierarchically for each of the three partitions:
  ```
  write_partitions -dir try3 TOPCHIP_SP sheet17 sheet25 sheet7
  ```
- The following command saves the partition information/files in the OpenAccess database format. The information for the top and the block level designs (all blocks) will be written in the `libForOA` directory view with the name `ptnView1`. It saves partitions without generating a floorplan file or a net list for the partitioned design. It also does not write the standard cells or trialRoute wires in the DEF file.
  ```
  write_partitions -partition_lib libForOA -partition_view ptnView1 -no_floorplan -no_netlist -no_std_cells -no_trialroute
  ```

Related Information

- `assemble_design`
- `read_db`
- `write_def`
- `commit_partitions`
- `set_module_model`
- `partition_floorplan` Category Attributes
- Partitioning the Design chapter in the *Innovus Stylus Common UI User Guide*
  - Saving Partitions

25

Placement Commands and Attributes

- add_decomp_filler
- add_endcaps
- add_endcaps Category Attributes
- add_filler_gaps
- add_fillers
- add_fillers Category Attributes
- add_gate_array_filler
- add_mim_cap
- add_spare_insts
- add_tap_walls
- add_well_taps
- add_well_taps Category Attributes
- check_filler
- check_library
- check_place
- create_gate_array_filler_group
- create_inst_space_group
- create_pg_keep_out
- create_pipeline_bus_guide
- create_pipeline_net_group
- create_scan_chain
- create_scan_chain_partition
- create_spare_module

- delete_all_cell_padding
- delete_all_scan_cells
- delete_base_cell_edge_type
- delete_cell_edge_spacing
- delete_cell_padding
- delete_cell_stack_area
- delete_cell_stack_group
- delete_cell_virtual_align
- delete_filler
- delete_gate_array_filler_group
- delete_inst_padding
- delete_inst_space_group
- delete_mim_cap
- delete_pg_keep_out
- delete_pipeline_net_group
- delete_scan_cell
- delete_scan_chain
- delete_scan_chain_partition
- delete_spare_module
- gui_clear_scan_chains
- gui_clear_spare_cells
- gui_show_scan_chains
- gui_show_spare_cells
- place_clones
- place_connected
- place_design
- place_detail
- place_fix_congestion

- place_inst
- place_jtag
- place_opt_design
- place_pad
- place_pipelines
- place_spare_modules
- place Category Attributes
- place Category Object Attributes
- reorder_scan
- reorder_scan Category Attributes
- report_cell_edge_spacing
- report_cell_edge_type
- report_cell_padding
- report_cell_stack_area
- report_cell_stack_group
- report_cell_virtual_align
- report_congestion
- report_density_in_area
- report_density_map
- report_inst_padding
- report_inst_space_group
- report_jtag_inst
- report_pg_keep_out
- report_pin_density
- report_pin_density_map
- report_pin_density Category Attributes
- report_pipelines
- report_place_density Category Attributes

- report_scan_cells
- report_scan_chain
- report_scan_chain_partitions
- reset_jtag
- reset_selective_blockage_gate
- reset_spare_insts
- set_cell_edge_spacing
- set_cell_edge_type
- set_cell_padding
- set_cell_stack_area
- set_cell_stack_group
- set_cell_virtual_align
- set_inst_padding
- set_jtag
- set_jtag_to_unplaced
- set_lockup_element
- set_scan_cell
- set_selective_blockage_gate
- set_spare_insts
- swap_well_taps
- trace_jtag
- trace_scan
- update_inbound_cell_site
- update_pipeline_net_group
- update_vertical_cell_edge_constraint
- pack_align_macros

add_decomp_filler

```
add_decomp_filler
[-help]
-bottom_edge_endcap {cell_name}
-core_column_cell {cell_name}
-core_row_cell {cell_name_list}
[-offset_x offset_in_microns]
[-offset_y offset_in_microns]
[-power_domain power_domain_name]
[-prefix prefix_name]
-step_x step_in_microns
-step_y step_in_microns
-top_edge_endcap {cell_name}
-vertical_edge_endcap {cell_name}
```

Inserts breaker cells to divide the chip into several small parts. This reduces the run time of coloring while manufacturing the chip.

This command is run after `create_floorplan` and before `add_endcaps`.

Parameters

`-bottom_edge_endcap {cell_name}`	Specifies the CA-break end cap cell for rows, to make bottom_edge of endcap face core. *Data_type*: string, required
`-core_column_cell {cell_name}`	Specifies the CA-break filler cell for inserting into columns. *Data_type*: string, required
`-core_row_cell {cell_name_list}`	Specifies a list of CA-break filler cells inserted horizontally in the core rows. *Data_type*: string, required
`-help`	Outputs a brief description that includes type and default information for each `add_decomp_filler` parameter. For a detailed description of the command and all of its parameters, use the man command: `man add_decomp_filler`.

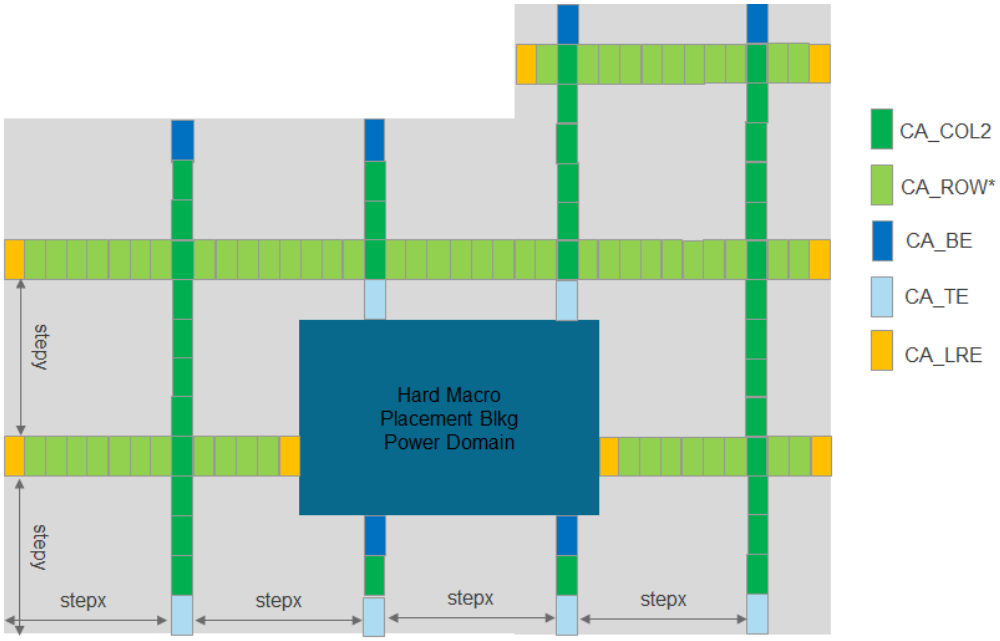
`-offset_x` *offset_in_microns*	Specifies the offset added to the first step in the horizontal direction. *Data_type*: float, required
`-offset_y` *offset_in_microns*	Specifies the offset added to the first step in the vertical direction. *Data_type*: float, required
`-power_domain` *power_domain_name*	Specifies the power domain for which CA-break filler cells are to be inserted. *Data_type*: string, optional **Note**: If the `-power_domain` option is used, the offset value is applied to the specified power domain. However, if the -power domain is not specified, the offset value is applied to all available power domains. To use different offset value for a specific power domain, you first need to delete the CA fillers from the power domain if it is already filled and then reinsert CA filler to the power domain with the target offset value.
`-prefix` *prefix_name*	Specifies the prefix of decomp cells. *Data_type*: string, optional
`-step_x` *step_in_microns*	Specifies the decomposition step in horizontal direction. *Data_type*: float, required
`-step_y` *step_in_microns*	Specifies the decomposition step in vertical direction. *Data_type*: float, required
`-top_edge_endcap {`*cell_name*`}`	Specifies CA-break end cap cell for rows, to make top_edge of endcap face core. *Data_type*: string, required
`-vertical_edge_endcap {`*cell_name*`}`	Specifies CA-break end cap cell inserted at the left and right edges as end caps. *Data_type*: string, required

Example

The code below shows the usage of the `add_decomp_filler` command.

```
add_decomp_filler
-core_row_cell {CA_ROW2 CA_ROW3} -core_column_cell {CA_COL2}
-top_edge_endcap CA_TE -bottom_edge_endcap CA_BE
-vertical_edge_endcap CA_LRE
-step_x 500 -step_y 500
```

Below is the graphical representation of the above example:



Related Information

- `create_floorplan`
- `add_endcaps`

add_endcaps

```
add_endcaps
[-help]
[-area x1 y1 x2 y2]
[-core_boundary_only]
[-power_domain powerDomainName]
[-prefix prefixName]
```

Places physical-only end cap cells at the ends of the site rows. A single-height cap cell is required at the end of each row. For deep N-well technology, the command can be used to insert row-caps to isolate p-well from top and bottom. Use `add_endcaps Category Attributes` to specify the row-cap cells. For deep N-well technology, `add_endcaps` also gets the end cap list from LEF if the end cap cell is specified with the following MACRO CLASS, but user definition has higher priority than LEF specification.

```
[CLASS ENDCAP {
| TOPEDGE | BOTTOMEDGE | LEFTEDGE | RIGHTEDGE
| LEFTTOPEDGE | RIGHTTOPEDGE
| LEFTBOTTOMEDGE | RIGHTBOTTOMEDGE
| LEFTTOPCORNER | RIGHTTOPCORNER
| LEFTBOTTOMCORNER | RIGHTBOTTOMCORNER}
} ]
```

Note: `add_endcaps` is always called after floorplan is done and before `add_well_taps`.

Note: End cap insertion honors core boundary, power domain boundary, placement blockage, fixed macro, and fixed macro halo

To delete the end cap cells, use the `delete_filler` command.

Note: This command is not meant for the insertion of multiple-height cells.

Parameters

`-area x1 y1 x2 y2`	Specifies the coordinates of the bounding box, in microns. By default, the core box that encloses all core rows is used. *Data_type*: rect, optional
`-core_boundary_only`	Specifies that end-cap cells are placed only within the core boundary. This ensures that end-cap cells do not get placed at locations where site rows are obstructed by blockages. *Data_type*: bool, optional
`-help`	Outputs a brief description that includes type and default information for each `add_endcaps` parameter. For a detailed description of the command and all of its parameters, use the man command: `man add_endcaps`.
`-power_domain` *`powerDomainName`*	Specifies the power domain in which the end-cap cells are inserted. This identifies the area where to place the end-cap cells and connect to the correct power and ground nets. *Data_type*: string, optional
`-prefix` *`prefixName`*	Specifies the prefix for the placed instances. The generated name for the instance uses this prefix as `prefix_`*`uniqueString`*. *Prefix Default:* `ENDCAP` *Data_type*: string, optional ⓘ Do not use an underscore at the end of the prefix name.

Note: If the given leftEdge cell is the same as the rightEdge cell, and the cell and site symmetries allow it, the postCap instances will be flipped in the Y direction.

Related Topics

- `add_endcaps Category Attributes`
- `add_well_taps`
- `delete_filler`
- Placing the Design chapter in the *Innovus Stylus Common UI User Guide*
    - "Adding End-Cap Cells"

add_endcaps Category Attributes

The root attributes in the `add_endcaps` category are:

- add_endcaps_avoid_two_sites_cell_abut
- add_endcaps_bottom_edge
- add_endcaps_boundary_tap
- add_endcaps_boundary_tap_swap_flow
- add_endcaps_cells
- add_endcaps_create_rows
- add_endcaps_double_height_incorner_protrusion
- add_endcaps_flip_y
- add_endcaps_incremental_left_edge
- add_endcaps_incremental_right_edge
- add_endcaps_left_bottom_corner
- add_endcaps_left_bottom_corner_even
- add_endcaps_left_bottom_corner_neighbor
- add_endcaps_left_bottom_corner_odd
- add_endcaps_left_bottom_edge
- add_endcaps_left_bottom_edge_even
- add_endcaps_left_bottom_edge_neighbor
- add_endcaps_left_bottom_edge_odd
- add_endcaps_left_edge
- add_endcaps_left_edge_bottom_border
- add_endcaps_left_edge_even
- add_endcaps_left_edge_odd
- add_endcaps_left_edge_top_border
- add_endcaps_left_top_corner
- add_endcaps_left_top_corner_even

- add_endcaps_left_top_corner_neighbor
- add_endcaps_left_top_corner_odd
- add_endcaps_left_top_edge
- add_endcaps_right_corner_bottom_border
- add_endcaps_right_corner_top_border
- add_endcaps_left_top_edge_even
- add_endcaps_left_top_edge_neighbor
- add_endcaps_left_top_edge_odd
- add_endcaps_min_horizontal_channel_width
- add_endcaps_min_jog_height
- add_endcaps_min_jog_width
- add_endcaps_min_vertical_channel_width
- add_endcaps_prefix
- add_endcaps_right_bottom_corner
- add_endcaps_right_bottom_corner_even
- add_endcaps_right_bottom_corner_neighbor
- add_endcaps_right_bottom_corner_odd
- add_endcaps_right_bottom_edge
- add_endcaps_right_bottom_edge_even
- add_endcaps_right_bottom_edge_neighbor
- add_endcaps_right_bottom_edge_odd
- add_endcaps_right_edge
- add_endcaps_right_edge_bottom_border
- add_endcaps_right_edge_even
- add_endcaps_right_edge_odd
- add_endcaps_right_edge_top_border
- add_endcaps_right_top_corner
- add_endcaps_right_top_corner_even

- add_endcaps_right_top_corner_neighbor
- add_endcaps_right_top_corner_odd
- add_endcaps_right_top_edge
- add_endcaps_right_top_edge_even
- add_endcaps_right_top_edge_neighbor
- add_endcaps_right_top_edge_odd
- add_endcaps_left_corner_bottom_border
- add_endcaps_left_corner_top_border
- add_endcaps_top_bottom_edge
- add_endcaps_top_edge
- add_endcaps_use_even_odd_sites
- add_endcaps_wall_keepout_from_vertical_boundary
- add_endcaps_wall_offset
- add_endcaps_wall_pitch
- add_endcaps_wall_to_convex_corner_spacing

To set the value of any root attribute, use the `set_db` command:

```
set_db attribute_name value
```

To get the current value of any root attribute, use the `get_db` command:

```
get_db attribute_name
```

To get the current value of all attributes in the `add_endcaps` category, use the following command:

```
get_db -category add_endcaps
```

add_endcaps_avoid_two_sites_cell_abut

`add_endcaps_avoid_two_sites_cell_abut {true | false}`

Default: `false`

Data_type: bool, read/write

Specifies to avoid two sites boundary cell abut to incorner cell in N10.

add_endcaps_bottom_edge

`add_endcaps_bottom_edge` *`ListOfCellNames`*

Default: ""

Data_type: string, read/write

Specifies the list of cells that have n-well cap on their bottom-edge when in `r0` orientation. Thus, these cells are used to cap a top-core row, or row at the bottom of a hard macro.

add_endcaps_boundary_tap

`add_endcaps_boundary_tap {true|false}`

Default: `false`

Data_type: bool, read/write

Specifies the boundary tap.

add_endcaps_boundary_tap_swap_flow

`add_endcaps_boundary_tap_swap_flow {true|false}`

Default: `true`

Data_type: bool, read/write

Specifies the boundary tap swap flow.

add_endcaps_cells

`add_endcaps_cells` *`ListOfCellNames`*

Default: `""`

Data_type: string, read/write

Specifies end cap cell candidates for N10 that are defined in LEF.

add_endcaps_create_rows

`add_endcaps_create_rows {true|false}`

Default: `false`

Data_type: bool, read/write

Creates rows for endtap cell techSites which do not have rows in the floorplan.

add_endcaps_double_height_incorner_protrusion

`add_endcaps_double_height_incorner_protrusion {true|false}`

Default: `false`

Data_type: bool, read/write

Allows the double height in-corner cell to protrude into core area.

add_endcaps_flip_y

`add_endcaps_flip_y {true|false}`

Default: `false`

Data_type: bool, read/write

Flips the orientation of the end cap instances in Y direction if the site symmetry allows it.

add_endcaps_incremental_left_edge

`add_endcaps_incremental_left_edge` *`ListOfCellNames`*

Default: ""

Data_type: string, read/write

Specifies the end cap lists with n-well at the left edge.

add_endcaps_incremental_right_edge

`add_endcaps_incremental_right_edge` *`ListOfCellNames`*

Default: ""

Data_type: string, read/write

Specifies the list of additional cells which should be treated as end cap by `check_endcaps` when checking the left edge of the core rows or rows starting at right edge of a hard macro.

Related Commands

`check_endcaps`

add_endcaps_left_bottom_corner

`add_endcaps_left_bottom_corner` *`cellName`*

Default: ""

Data_type: string, read/write

Specifies the cell which has n-well cap on its left-bottom corner when in `r0` orientation. These cells are added on inner corners of core, or concave corner of a rectilinear hard macro.

add_endcaps_left_bottom_corner_even

`add_endcaps_left_bottom_corner_even` *`cellName`*

Default: ""

Data_type: string, read/write

Specifies end cap with, n-well at the left-bottom corner, even sites.

add_endcaps_left_bottom_corner_neighbor

`add_endcaps_left_bottom_corner_neighbor` *`cellName`*

Default: ""

Data_type: string, read/write

Specifies the cell to be abutted with the left side of leftBottomCorner in `r0` orientation.

add_endcaps_left_bottom_corner_odd

`add_endcaps_left_bottom_corner_odd` *`cellName`*

Default: ""

Data_type: string, read/write

Specifies end cap with n-well at the left-bottom corner, odd sites.

add_endcaps_left_bottom_edge

`add_endcaps_left_bottom_edge` *`cellName`*

Default: ""

Data_type: string, read/write

Specifies the one cell that has n-well cap on its left-bottom edge when in `r0` orientation. Thus, this cell, in `r0` orientation, can be used to cap a row ending at left-bottom convex corner of a hard macro, or rectilinear core; and if allowed to flip in Y, it can be used on `my` rows that end in left-top convex corner. With both X and Y symmetry, it can cap all convex corners, though only `r0` rows at bottom, and `my` rows for top corners.

add_endcaps_left_bottom_edge_even

`add_endcaps_left_bottom_edge_even` *`cellName`*

Default: ""

Data_type: string, read/write

Specifies end cap with n-well at the left-bottom edge, even sites.

add_endcaps_left_bottom_edge_neighbor

`add_endcaps_left_bottom_edge_neighbor` *`cellName`*

Default: ""

Data_type: string, read/write

Specifies the cell to be abutted with the right edge of leftBottomEdge in the `r0` orientation.

add_endcaps_left_bottom_edge_odd

`add_endcaps_left_bottom_edge_odd` *`cellName`*

Default: ""

Data_type: string, read/write

Specifies end cap with n-well at the left-bottom edge, odd sites.

add_endcaps_left_edge

`add_endcaps_left_edge` *`cellName`*

Default: ""

Data_type: string, read/write

Specifies the one cell that has n-well cap on its left edge when in `r0` orientation. Thus, this cell is used to cap right edge of the core rows, or rows ending at left edge of a hard macro. This would be same as a post-cap, in a two well process.

add_endcaps_left_edge_bottom_border

`add_endcaps_left_edge_bottom_border` *`cellList`*

Default: ""

Data_type: string, read/write

Specifies end cap with n-well at the left edge, poly at the bottom boundary.

add_endcaps_left_edge_even

`add_endcaps_left_edge_even` *`cellName`*

Default: ""

Data_type: string, read/write

Specifies end cap with n-well at the left edge, even sites.

add_endcaps_left_edge_odd

`add_endcaps_left_edge_odd` *`cellName`*

Default: ""

Data_type: string, read/write

Specifies end cap with n-well at the left edge, odd sites.

add_endcaps_left_edge_top_border

`add_endcaps_left_edge_top_border` *`cellList`*

Default: ""

Data_type: string, read/write

Specifies end cap with n-well at the right edge, poly at the top boundary.

add_endcaps_left_top_corner

`add_endcaps_left_top_corner` *cellName*

Default: `""`

Data_type: string, read/write

Specifies the cell that has n-well cap on its left top corner when in `r0` orientation. Thus, this cell is added on inner corners of core, or concave corner of a rectilinear hard macro.

add_endcaps_left_top_corner_even

`add_endcaps_left_top_corner_even` *cellName*

Default: `""`

Data_type: string, read/write

Specifies end cap with n-well at the left-top corner, even sites.

add_endcaps_left_top_corner_neighbor

`add_endcaps_left_top_corner_neighbor` *cellName*

Default: ""

Data_type: string, read/write

Specifies the cell to be abutted with the left side of leftTopCorner in `r0` orientation.

add_endcaps_left_top_corner_odd

`add_endcaps_left_top_corner_odd` *cellName*

Default: `""`

Data_type: string, read/write

Specifies end cap with n-well at the left-top corner, odd sites.

add_endcaps_left_top_edge

`add_endcaps_left_top_edge` *`cellName`*

Default: `""`

Data_type: string, read/write

Specifies the cell that has n-well cap on its top-left edge when in `r0` orientation. Thus, in `r0` orient, it is the flipped-in-Y version of left-bottom edge cell.

add_endcaps_right_corner_bottom_border

`add_endcaps_right_corner_bottom_border` *`ListOfCellNames`*

Default: ""

Data_type: string, read/write

Specifies the inner right corner bottom border cell.

add_endcaps_right_corner_top_border

`add_endcaps_right_corner_top_border` *`ListOfCellNames`*

Default: ""

Data_type: string, read/write

Specifies the inner right corner top border cell.

add_endcaps_left_top_edge_even

`add_endcaps_left_top_edge_even` *`cellName`*

Default: `""`

Data_type: string, read/write

Specifies end cap with n-well at the left-top edge, even sites.

add_endcaps_left_top_edge_neighbor

`add_endcaps_left_top_edge_neighbor` *`cellName`*

Default: `""`

Data_type: string, read/write

Specifies the cell to be abutted with the right edge of leftTopEdge in the `r0` orientation.

add_endcaps_left_top_edge_odd

`add_endcaps_left_top_edge_odd` *`cellName`*

Default: `""`

Data_type: string, read/write

Specifies end cap with n-well at the left-top edge, odd sites.

add_endcaps_min_horizontal_channel_width

`add_endcaps_min_horizontal_channel_width` *`channelWidth`*

Default: `0`

Data_type: int, read/write

If horizontal channel width is smaller than specified value, addEndCap will cover this channel with hard blockage.

add_endcaps_min_jog_height

`add_endcaps_min_jog_height` *`jogHeight`*

Default: `0`

Data_type: int, read/write

If jog height is smaller than specified value, addEndCap will cover this jog with hard blockage.

add_endcaps_min_jog_width

`add_endcaps_min_jog_width` *`jogWidth`*

Default: `0`

Data_type: int, read/write

If jog width is smaller than specified value, addEndCap will cover this jog with hard blockage.

add_endcaps_min_vertical_channel_width

`add_endcaps_min_vertical_channel_width` *`channelWidth`*

Default: 0

Data_type: int, read/write

If vertical channel width is smaller than specified value, addEndCap will cover this channel with hard blockage.

add_endcaps_prefix

`add_endcaps_prefix` *`endcap-prefix`*

Default: `""`

Data_type: string, read/write

Adds prefix of EndCap cells.

add_endcaps_right_bottom_corner

`add_endcaps_right_bottom_corner` *`cellName`*

Default: ""

Data_type: string, read/write

Specifies the cell that has n-well cap on its right bottom corner when in `r0` orientation. Thus, this cell is added on inner corners of core, or concave corner of a rectilinear hard macro.

add_endcaps_right_bottom_corner_even

`add_endcaps_right_bottom_corner_even` *`cellName`*

Default: ""

Data_type: string, read/write

Specifies end cap with n-well at the right-bottom corner, even sites.

add_endcaps_right_bottom_corner_neighbor

`add_endcaps_right_bottom_corner_neighbor` *`cellName`*

Default: ""

Data_type: string, read/write

The cell to be abutted with the right side of rightBottomCorner in `r0` orientation.

add_endcaps_right_bottom_corner_odd

`add_endcaps_right_bottom_corner_odd` *`cellName`*

Default: ""

Data_type: string, read/write

Specifies end cap with n-well at the right-bottom corner, odd sites.

add_endcaps_right_bottom_edge

`add_endcaps_right_bottom_edge` *`cellName`*

Default: ""

Data_type: string, read/write

Specifies the cell that has n-well cap on its right bottom edge when in `r0` orientation. Thus, these cells are the flipped-in-X version of add_endcaps_left_bottom_edge cell. Usually one of these 2 is sufficient for a library.

add_endcaps_right_bottom_edge_even

`add_endcaps_right_bottom_edge_even` *`cellName`*

Default: ""

Data_type: string, read/write

Specifies end cap with n-well at the right-bottom edge, even sites.

add_endcaps_right_bottom_edge_neighbor

`add_endcaps_right_bottom_edge_neighbor` *`cellName`*

Default: ""

Data_type: string, read/write

Specifies the cell to be abutted with the left edge of rightBottomEdge in the `r0` orientation.

add_endcaps_right_bottom_edge_odd

`add_endcaps_right_bottom_edge_odd` *`cellName`*

Default: ""

Data_type: string, read/write

Specifies end cap with n-well at the right-bottom edge, odd sites.

add_endcaps_right_edge

`add_endcaps_right_edge` *`cellName`*

Default: ""

Data_type: string, read/write

Specifies the cell that has n-well cap on its right edge when in `r0` orientation. Thus, these cells are used to cap the left edge of the core-rows in `r0`. In 2 well process, this would be the pre-cap cell.

add_endcaps_right_edge_bottom_border

`add_endcaps_right_edge_bottom_border` *`cellList`*

Default: ""

Data_type: string, read/write

Specifies end cap with n-well at the right edge, poly at the bottom boundary.

add_endcaps_right_edge_even

`add_endcaps_right_edge_even` *`cellName`*

Default: ""

Data_type: string, read/write

Specifies end cap with n-well at the right edge, even sites.

add_endcaps_right_edge_odd

`add_endcaps_right_edge_odd` *`cellName`*

Default: ""

Data_type: string, read/write

Specifies end cap with n-well at the right edge, odd sites.

add_endcaps_right_edge_top_border

`add_endcaps_right_edge_top_border` *`cellList`*

Default: ""

Data_type: string, read/write

Specifies end cap with n-well at the right edge, poly at the top boundary.

add_endcaps_right_top_corner

`add_endcaps_right_top_corner` *`cellName`*

Default: ""

Data_type: string, read/write

Specifies the cell that has n-well cap on their right top corner when in `r0` orientation. Thus, this cell, in `r0`, is added on right corner of core, or concave corner of a rectilinear hard macro.

add_endcaps_right_top_corner_even

`add_endcaps_right_top_corner_even` *`cellName`*

Default: ""

Data_type: string, read/write

Specifies end cap with n-well at the right-top corner, even sites.

add_endcaps_right_top_corner_neighbor

`add_endcaps_right_top_corner_neighbor` *`cellName`*

Default: ""

Data_type: string, read/write

Specifies the cell to be abutted with the right side of rightTopCorner in `r0` orientation.

add_endcaps_right_top_corner_odd

`add_endcaps_right_top_corner_odd` *`cellName`*

Default: ""

Data_type: string, read/write

Specifies end cap with n-well at the right-top corner, odd sites.

add_endcaps_right_top_edge

`add_endcaps_right_top_edge` *`cellName`*

Default: ""

Data_type: string, read/write

Specifies the cell that has n-well cap on its right top edge when in `r0` orientation. Thus, this cell is the `r180` version of add_endcaps_left_bottom_edge.

add_endcaps_right_top_edge_even

`add_endcaps_right_top_edge_even` *`cellName`*

Default: ""

Data_type: string, read/write

Specifies end cap with n-well at the right-top edge, even sites.

add_endcaps_right_top_edge_neighbor

`add_endcaps_right_top_edge_neighbor` *`cellName`*

Default: ""

Data_type: string, read/write

Specifies the cell to be abutted with the left edge of rightTopEdge in the `r0` orientation.

add_endcaps_right_top_edge_odd

`add_endcaps_right_top_edge_odd` *`cellName`*

Default: ""

Data_type: string, read/write

Specifies end cap with n-well at the right-top edge, odd sites.

add_endcaps_left_corner_bottom_border

`add_endcaps_left_corner_bottom_border` *`ListOfCellNames`*

Default: `""`

Data_type: string, read/write

Specifies the inner left corner, bottom border cell.

add_endcaps_left_corner_top_border

`add_endcaps_left_corner_top_border` *`ListOfCellNames`*

Default: `""`

Data_type: string, read/write

Specifies the inner left corner, top border cell.

add_endcaps_top_bottom_edge

`add_endcaps_top_bottom_edge` *`cellName`*

Default: ""

Data_type: string, read/write

Specifies the cell that has an n-well cap on its top edge as well as bottom edge when in `r0` orientation.

add_endcaps_top_edge

`add_endcaps_top_edge` *`cellName`*

Default: ""

Data_type: string, read/write

Specifies the list of cells that have n-well cap on their top edge when in `r0` orientation. Thus, these cells are used to cap the bottom-core row, or row at top of a hard macro.

add_endcaps_use_even_odd_sites

`add_endcaps_use_even_odd_sites {none|even|odd}`

Default: `none`

Data_type: enum {none even odd}, read/write

Specifies to use even or odd poly sites.

- `none`: EVEN/ODD cells are not considered even if they are specified
- `even`: Need to specify EVEN boundary cell
- `odd`: Need to specify ODD boundary cell

add_endcaps_wall_keepout_from_vertical_boundary

add_endcaps_wall_keepout_from_vertical_boundary `{true|false}`

Default: 0.0

Data_type: double, read/write

Specifies the wall keep-out spacing from left/right boundary cell.

add_endcaps_wall_offset

`add_endcaps_wall_offset` *macro*

Default: 0.0

Data_type: double, read/write

Specifies the x coordinate of the first wall column.

add_endcaps_wall_pitch

`add_endcaps_wall_pitch` *macro*

Default: 0.0

Data_type: double, read/write

Specifies the wall pitch.

add_endcaps_wall_to_convex_corner_spacing

`add_endcaps_wall_to_convex_corner_spacing` *macro*

Default: 0.0

Data_type: double, read/write

Constraints the spacing between wall and convex corner.

add_filler_gaps

```
add_filler_gaps
min_gap
[-effort {none | low | medium | high}]
[-radius value_in_micron]
```

Moves placed standard cell instances to create gaps so that filler cells can be added. Tries to move cells to create a gap of the specified size. If enough space is not available, moves cells until they touch each other. If a gap already exists, and it is larger than the specified size, the command does not move cells to make it smaller.

By default, this command deletes all routing, because the routing is no longer valid due to the movement of the cells. To override this behavior, specify the following command:

```
set_db place_detail_preserve_routing true
```

This command has the following limitations:

- It does not moved `FIXED` cells.
- It does not guarantee to that all gaps will be larger than the specified minimum gap size.

Use this command after placement and before routing.

Parameters

min_gap	Specifies the minimum-size gap in which a filler cell can be added. Unit is micron. *Data_type*: float, required
`-effort {none | low| medium | high}`	Specifies the effort the command needs to spend for honoring the given gap constraint. • `none` effort specifies that no standard cell is moved for honoring the giving gap constraint. • `low` effort specifies that only single height standard cells can be moved within the same row. • `medium` effort specifies that only single height standard cells can be moved across the rows. • `high` effort specifies that the standard cells can be moved across rows, and double height standard cells can be moved within the same row. *Default*: `low` *Data_type*: enum, optional
`-radius` *value_in_micron*	Specifies the searching radius when solving the minimum filler gap. *Default*: 16 *Data_type*: float, optional

Example

- The following command moves standard cells within a row to create gaps of at least two microns between the cells:
  ```
  add_filler_gaps 0.168
  ```
- The following command specifies two site widths with each side width equal to `[lindex [lsort [get_db sites.size_x -u ] 0]`.
  ```
  add_filler_gaps [expr 2 * [lindex [ lsort [get_db sites.size_x -u ] 0]]
  ```

Related Information

- place Category Attributes
- set_db

add_fillers

```
add_fillers
[-area  x1   y1   x2   y2 ]
[-base_cells { filler_cell_name_list }]
[-create_rows {true | false}]
[-check_different_cells {true|false}]
[-check_drc {true | false}]
[-check_via_enclosure {true | false}]
[-density  targetUtilization ]
[-eco_mode {true | false}]
[-fill_gap]
[-fix_drc]
[-fix_horizontal_max_length_violation]
[-fix_vertical_max_length_violation]
[-honor_preroute_as_obs {true | false}]
[-mark_fixed]
[-check_min_hole {true | false}]
[-merge {true | false}]
[-power_domain powerDomainName ]
[-prefix prefixName ]
[-prevent_vertical_stack_max_length_violation]
```

Inserts filler cell instances in the gaps between standard cell instances. Filler cell instances provide continuity for the power and ground rails, as well as for n-wells. In most cases, use this command after placement and before detailed routing. If the design is routed, this command also does DRC checks of the filler cells added versus the wires in the design. It does not check versus adjacent cells. This command is fast enough to be used many times in the design flow.

Note: Before using `add_fillers`, run the `connect_global_net` command to provide global-net-connection rules for supply pins of the added fillers. Without these rules, the built-in design-rule checks of `add_fillers` will not be accurate.

To resolve adjacent instance DRC violations created by the insertion of filler cell instances, run the `verify_drc` command to mark these violations. Then, use the `add_fillers` command again with the `-fix_drc` parameter to replace the filler cell instances that are in violation.

Note: This command resolves only filler-related violations; it cannot resolve net-based violations. To repair net-based violations, reroute the violated net(s).

The `add_fillers` command recognizes up to eight implant layers and attempts to honor implant layer width and spacing rules within each implant layer. If it does not find the correct filler cell to add to an implant layer, it tries to add "no-implant" filler cells. If it cannot add the "no-implant" cells, it inserts a filler cell that is large enough that it does not violate the minimum width rule of the relevant implant

layer.

Use the `delete_filler` command to remove the added physical instances, including filler cell instances .

Note: This command supports multithreading.

> Some `add_fillers` parameters can also be specified with add_fillers Category Attributes. However, the parameters specified with `add_fillers` command apply to the current `add_fillers` invocation only, and supersede the parameters set by `add_fillers` attributes. For more information, see add_fillers Category Attributes.

Parameters

`-area` *x1 y1 x2 y2*	Adds filler cells to the specified area. If the area is not defined properly, does not add filler cells. Normally, designs are placed before filler cells are added. However, If you use this parameter and the design is not placed, you get a warning, and the command adds fillers in the specified area. • ***x1*** specifies the area's lower left point on the x axis. • ***y1*** specifies the area's lower left point on the y axis. • ***x2*** specifies the area's upper right point on the x axis. • ***y2*** specifies the area's upper right point on the y axis. *Data_type*: rect, optional

`-base_cells` *`filler_cell_name_list`*	Specifies the list of filler base cell names to add. Separate names with spaces and enclose the list with quotation marks (") or curly braces ({}). *Data_type*: string, optional
`-create_rows {true | false}`	Controls whether `add_fillers` creates rows in the floorplan if a cell's site does not have rows. *Default*: `true` *Data_type*: bool, optional
`-check_different_cells {true|false}`	Checks violations between different cell types. *Default*: `false` *Data_type*: bool, optional **Note**: Using this feature will increase the runtime of `add_fillers`.

`-check_drc {true | false}`	Controls postroute DRC violation checking against existing net routing, without considering adjacent cell DRC violations. This parameter does not disable regular placement checks. *Default*: `true` *Data_type*: bool, optional If filler cells are added after routing, the software checks for DRC violations between regular nets and the filler cells. When the inserted filler violates the existing routing, `add_fillers` tries to use other filler cells to replace the violating one; if it fails to fill the gap without violation, it leaves a gap. In 16.10, add_fillers has been enhanced with DRC. This flow would fill gaps as much as possible even with DRC, and will give the violation report and marker for easy debug. Also, the `-add_fillers_with_drc` attribute calls second add_fillers without DRC checking. If filler cells are added after routing, the software checks for DRC violations between regular nets and the filler cells. When the inserted filler violates the existing routing, add_fillers tries to use other filler cells to replace the violating one. If it fails to fill the gap without violation, it will insert a violated filler in the second pass and give the violation report and marker for easy debug when `-add_fillers_with_drc` attribute is turned on. Note: The setting of `place_detail_use_check_drc` attribute can affect `add_fillers` ability to add instances without causing DRC violations.

`-eco_mode {true | false}`	Controls whether the software resolves overlaps of standard cells with fillers by removing the overlapping fillers, and inserting new fillers in any existing or newly created gaps due to partial overlaps in eco flow. If this parameter is used after routing, the command also resolves DRC violations of filler geometries of all fillers with regular wires. *Default*: `false` *Data_type*: bool, optional
`-fill_gap`	Fills a gap between cells by adding a combination of cells, instead of by adding the single largest cell that fits, if doing so avoids leaving an unfilled single-width gap. *Data_type*: bool, optional
`-fix_drc`	Corrects DRC violations reported by `verify_drc` between filler cells and adjacent standard cells. *Data_type*: bool, optional **Note**: The `add_fillers` command resolves only filler-related violations; it cannot resolve net-based violations. To repair net-based violations, reroute the violated net(s). Use this parameter in subsequent runs of the `add_fillers` command, after adding filler cells and checking for violations with `verify_drc`. The command replaces filler cells that cause DRC violations with fillers that do not cause violations. If it cannot replace a violating filler cell without causing another violation, it leaves a gap.
`-fix_horizontal_max_length_violation`	Fixes horizontal max length violations. *Data_type*: bool, optional

`-fix_vertical_max_length_violation`	Fixes CPODE max length violations caused by filler cells. *Data_type*: bool, optional
`-honor_preroute_as_obs {true | false}`	Instructs `add_fillers` to honor the setting of the `place_detail_preroute_as_obs` attribute. When `honor_preroute_as_obs` is `true` and `place_detail_preroute_as_obs` is set, `add_fillers` treats preroutes on metal layers where filler cells have geometries and are enumerated by `place_detail_preroute_as_obs` as obstructions, and ignores preroutes on other layers. For example, if `place_detail_preroute_as_obs` is set to metal1 and metal2, and all the filler cell geometries are on metal1, `add_fillers` treats metal1 preroutes as obstructions but not metal2 preroutes. *Default*: `false` *Data_type*: bool, optional
`-mark_fixed`	Mark inserted fillers as FIXED. *Data_type*: bool, optional
`-merge {true | false}`	When checking for spacing violations, merge same-net geometries to determine width-dependent spacing. If the LEF file contains width-dependent rules, this parameter causes `add_fillers` to have the following behavior: • **true:** merges overlapping same-net objects to create a large rectangle. • **false:** uses the unmerged same-net object shapes. *Default*: `false` *Data_type*: bool, optional

`-check_min_hole {true | false}`	Calls `verify_drc` to check for minimum hole violations before adding filler cells and repairs the violations when `add_fillers` runs. *Default*: `false` *Data_type*: bool, optional
`-power_domain` *powerDomainName*	Specifies the power domain in which to add filler cells. *Default*: tries to add fillers to all power domains. *Data_type*: string, optional
`-prefix` *prefixName*	Specifies the prefix for the placed instances. *Default*: `FILLER` *Data_type*: string, optional
`-density` *targetUtilization*	Specifies the percent of free space remaining after placement to fill with given filler cell instances. Use `-density` to add gate-array core filler cells to use later in the design flow, along with gate-array core cells, for metal-only mask changes. *Range*: More than `0.0` (`0%`) and equal to or less than `1.0` (`100%`) *Default*: `1.0` *Data_type*: float, optional
`-check_via_enclosure {true | false}`	Calls `verify_drc` to check for via enclosure violations before adding filler cells and repairs the violations when `add_fillers` runs. *Default*: `false` *Data_type*: bool, optional

`-prevent_vertical_stack_max_length_violation`	Used before endcap insertion, inserts larger encaps at left/right boundaries to fix CPODE max length violations. *Data_type*: bool, optional

Examples

- The following commands insert filler cell instances for cells `FILL64`, `FILL16`, `FILL4`, `FILL2`, and `FILL1`:

```
add_fillers -base_cells {FILL64 FILL16 FILL4 FILL2 FILL1}
add_fillers -base_cells "FILL64 FILL16 FILL4 FILL2 FILL1"
add_fillers -base_cells [ list FILL64 FILL16 FILL4 FILL2 FILL1 ]
add_fillers -base_cells FILL64 FILL16 FILL4 FILL2 FILL1
```

- The following commands show the use of the `-density` parameter to set the utilization of `GACORE` filler cells to 50 percent:

```
# Place the design
place_design
# Now fill with GACORE cells up to 50% of remaining space
# area utilization
add_fillers -base_cells {gcore16 gcore8 gcore4} -prefix GACORE -density 0.5
# Now fill with the standard filler cells
add_fillers -base_cells {fill8 fill4 fill2 fill} -prefix FILLER
# Continue with routing
```

- The following command enable postRoute DRC checking for min hole violations and via enclosure violations:

```
add_fillers -base_cells {FILL16 FILL8 FILL4 FILL2} -check_drc true -check_min_hole
true -check_via_enclosure true
```

Related Information

- add_fillers Category Attributes
- connect_global_net
- delete_filler
- place Category Attributes

add_fillers Category Attributes

The root attributes in the add_fillers category are:

- add_fillers_cells
- add_fillers_avoid_abutment_patterns
- add_fillers_cell_name_style
- add_fillers_check_different_cells
- add_fillers_check_drc
- add_fillers_create_rows
- add_fillers_distribute_implant_evenly
- add_fillers_eco_mode
- add_fillers_horizontal_exception_cell
- add_fillers_honor_preroute_as_obs
- add_fillers_keep_fixed
- add_fillers_horizontal_max_length
- add_fillers_horizontal_repair_cell
- add_fillers_no_single_site_gap
- add_fillers_prefix
- add_fillers_preserve_user_order
- add_fillers_scheme
- add_fillers_swap_cell
- add_fillers_vertical_stack_exception_cell
- add_fillers_vertical_stack_left_edge_exception_cell
- add_fillers_vertical_stack_max_length
- add_fillers_vertical_stack_repair_cell
- add_fillers_vertical_stack_repair_edge
- add_fillers_vertical_stack_right_edge_exception_cell
- add_fillers_with_drc

- add_fillers_y_flip_type

To set the value of any root attribute, use the `set_db` command:

```
set_db attribute_name value
```

To get the current value of any root attribute, use the `get_db` command:

```
get_db attribute_name
```

To get the current value of all attributes in the `add_fillers` category, use the following command:

```
get_db -category add_fillers
```

add_fillers_cells

`add_fillers_cells {{list-of-cells1} {list-of-cells2} ...}`

Default: `""`

Data_type: string, read/write

Specifies the filler cells to add with add_fillers. Enclose the filler cell names in quotation marks (") or curly braces ({ }). You can specify multiple cell lists at one time, using curly braces ({ }) to separate each one.

add_fillers_avoid_abutment_patterns

`add_fillers_avoid_abutment_patterns {{ 1:1 1:2 ...}}`

Default: `""`

Data_type: string, read/write

Specifies illegal abutment patterns.

add_fillers_cell_name_style

`add_fillers_cell_name_style {hier|flat}`

Default: `hier`

Data_type: enum {hier flat}, read/write

Adds physical cells into hierarchical modules, or as top level cells (flat).

add_fillers_check_different_cells

`add_fillers_check_different_cells {true | false}`

Default: `false`

Data_type: bool, read/write

Checks violations between different cell types.

Note: Using this feature will increase the runtime of `add_fillers`.

add_fillers_check_drc

`add_fillers_check_drc {true | false}`

Default: `true`

Data_type: bool, read/write

When `add_fillers` is called after routing, instructs the software to run DRC checks of filler pins with signal net wires. To disable the DRC checks, set this parameter to `false`.

add_fillers_create_rows

`add_fillers_create_rows {true | false}`

Default: `true`

Data_type: bool, read/write

Instructs `add_fillers` to create rows in the floorplan.

By default.if a cell's site does not have rows in the floorplan, `add_fillers` creates the rows.

add_fillers_distribute_implant_evenly

`add_fillers_distribute_implant_evenly {true | false}`

Default: `false`

Data_type: bool, read/write

Distributes implant evenly for fillers.

add_fillers_eco_mode

`add_fillers_eco_mode {true | false}`

Default: `false`

Data_type: bool, read/write

Controls whether the software resolves overlaps of standard cells with fillers by removing the overlapping fillers, and inserting new fillers in any existing or newly created gaps due to partial overlaps in eco flow. If this attribute is used after routing, the command also resolves DRC violations of filler geometries of all fillers with regular wires.

add_fillers_horizontal_exception_cell

`add_fillers_horizontal_exception_cell` *`list_of_cells`*

Default:

Data_type: string, read/write

Specifies the list of exceptional cells for horizontal stack checking.

add_fillers_honor_preroute_as_obs

`add_fillers_honor_preroute_as_obs {true | false}`

Default: `false`

Data_type: bool, read/write

Instructs `add_fillers` to honor the setting of `place_detail_preroute_as_obs`. By default, `add_fillers` does not consider any pre-Routes to be obstructions.

When this attribute is `1`, `add_fillers` treats pre-Routes on metal layers where filler cells have geometries as obstructions, and ignores pre-Routes on other layers. In most cases, this means that `add_fillers` treats pre-Routes on *metal1* as obstructions and ignores pre-Routes on other metal layers. For example, if `place_detail_preroute_as_obs` is set to *metal1* and *metal2*, and all the filler cell geometries are on *metal1*, `add_fillers` treats *metal1* pre-Routes as obstructions but does not treat *metal2* pre-Routes as obstructions.

`add_fillers` ignores `place_detail_preroute_as_obs` and inserts fillers in all legal sites.

add_fillers_keep_fixed

```
add_fillers_keep_fixed {true | false}
```

Default: `false`

Data_type: bool, read/write

Keeps the FIXED filler cells.

add_fillers_horizontal_max_length

```
add_fillers_horizontal_max_length instance_max_length
```

Default: `0.0`

Data_type: double, read/write

Specifies the instance max length (in micron) aligned in horizontal direction.

add_fillers_horizontal_repair_cell

```
add_fillers_horizontal_repair_cell list_of_cells
```

Default: `{}`

Data_type: string, read/write

Specifies the list of filler cells to fix horizontal max length violation.

add_fillers_no_single_site_gap

```
add_fillers_no_single_site_gap {true | false}
```

Default: `true`

Data_type: bool, read/write

Fills a gap between cells by adding a combination of cells, instead of by adding the single largest cell that fits, if doing so avoids leaving an unfilled single-width gap.

add_fillers_prefix

```
add_fillers_prefix prefix
```

Default: `""`

Data_type: string, read/write

Specifies the prefix for the added filler cells.

add_fillers_preserve_user_order

```
add_fillers_preserve_user_order {true | false}
```

Default: `false`

Data_type: bool, read/write

Specifies whether using the user defined filler cells order for filler insertion.

Filler cells in each core list are sorted from largest to smallest.

add_fillers_scheme

```
add_fillers_scheme {locationFirst | cellFirst}
```

Default: `locationFirst`

Data_type: enum {locationFirst cellFirst}, read/write

Specifies the filler insertion scheme.

- `locationFirst` fills each given location with the best filler before proceeding to next location. It always starts with the first empty site.
- `cellFirst` inserts filler cells based on the user-specified list. The tool tries to insert the first filler from the user list in all possible locations before moving to the second cell in the list.

add_fillers_swap_cell

```
add_fillers_swap_cell {{cell swap pair1} {cell swap pair2} ...}
```

Default: `""`

Data_type: string, read/write

Swaps the specified cell pairs.

add_fillers_vertical_stack_exception_cell

`add_fillers_vertical_stack_exception_cell` *`list_of_cells`*

Default: `{}`

Data_type: string, read/write

Specifies the list of exceptional cells for vertical stack checking.

add_fillers_vertical_stack_left_edge_exception_cell

`add_fillers_vertical_stack_left_edge_exception_cell` *`list_of_cells`*

Default: `""`

Data_type: string, read/write

Specifies a list of cells in which only the left edge will not be checked for vertical stack violation.

add_fillers_vertical_stack_max_length

`add_fillers_vertical_stack_max_length` *`list_of_cells`*

Default: `0.0`

Data_type: double, read/write

Specifies the instance max length (in micron) aligned in the vertical direction.

add_fillers_vertical_stack_repair_cell

`add_fillers_vertical_stack_repair_cell` *`list_of_cells`*

Default: `{}`

Data_type: string, read/write

Specifies the list of filler cells to fix vertical stack violation.

add_fillers_vertical_stack_repair_edge

`add_fillers_vertical_stack_repair_edge {left|right}`

Default: `left`

Data_type: enum {left right}, read/write

Specifies the edge to use of vertical repair cell.

add_fillers_vertical_stack_right_edge_exception_cell

`add_fillers_vertical_stack_right_edge_exception_cell` *`list_of_cells`*

Default: `""`

Data_type: string, read/write

Specifies a list of cells in which only the right edge will not be checked for vertical stack violation.

add_fillers_with_drc

`add_fillers_with_drc {true | false}`

Default: `true`

Data_type: bool, read/write

Calls the second addFiller without DRC checking.

add_fillers_y_flip_type

`add_fillers_y_flip_type { integer : { first | last } ... }`

Default: `""`

Data_type: string, read/write

Defines the first or the last cell with certain cell size should be y-flipped.

add_gate_array_filler

```
add_gate_array_filler
[-help]
[-area x1 y1 x2 y2]
[-allow_shift_distance x y]
[-cell_orientation {R0|MX|MY|R180}]
[-offset_x offset_in_microns]
[-offset_y offset_in_microns]
[-power_domain powerDomainName]
[-prefix prefixName]
{-cell cellName -step_x step_in_microns -step_y step_in_microns}
```

Inserts GA filler cells with regular pattern in the pre-place stage.

The GA filler cells created by this command are marked as physical-only cells. This means that the `isPhysOnly` attribute of these GA filler cells is set to `true`. This is a prerequisite for the `place_eco -use_gate_array_filler_cells` command to map the unplaced ECO cells to the added GA filler cells, in case of a GA filler-based post-mask ECO.

Parameters

`-allow_shift_distance x y`	Specifies the distance between the new and the original position in the x and y directions. If you do not specify `-allow_shift_distance`, the GA filler will be removed if it is added at original location with a violation. If `-allow_shift_distance` is specified, the GA can be shifted to avoid violation. If a GA filler cannot be added at the original location because of the violation, it will be moved to the left side, right side, upside, or downside of the original location to ensure that there is no violation. The next filler keeps the step value with the original position of the moved filler.
`-area x1 y1 x2 y2`	Specifies the area in which GA fillers are to be inserted. If no area is specified, GA fillers will be inserted for the complete core area. *Data_type*: rect, optional
`-cell cellName`	Specifies the name of the filler cell to be used. *Data_type*: string, optional

`-cell_orientation {R0|MX|MY|R180}`	Specifies all legal orientations of GFILL instances. *Data_type*: string, optional
`-offset_x` *offset_in_microns*	Specifies the offset, in microns, from the origin in the horizontal direction. If the `-area` parameter is specified, the origin is (x1, y1) of the box defined by `-area`. If `-area` is not specified the origin is (0,0). *Data_type*: float, optional
`-offset_y` *offset_in_microns*	Specifies the offset, in microns, from the origin in the vertical direction. If the `-area` parameter is specified, the origin is (x1, y1) of the box defined by `-area`. If `-area` is not specified the origin is (0,0). *Data_type*: float, optional
`-power_domain` *powerDomainName*	Specifies the power domain for which fillers are to be inserted. *Data_type*: string, optional
`-prefix` *prefixName*	Specifies the prefix for the inserted GA filler cells. *Data_type*: string, optional
`-step_x` *step_in_microns*	Specifies the distance between GA fillers in the horizontal direction. *Data_type*: float, optional
`-step_y` *step_in_microns*	Specifies the distance between GA fillers in the vertical direction. *Data_type*: float, optional

Examples

- The following command adds GA fillers of type `GAFILL4` with spacing in horizontal and vertical direction of 20 microns. The first GAFILL4 will be placed at (10, 0).

```
add_gate_array_filler -cell GAFILL4 -step_x 20 -step_y 20 -offsetx 10
```

- The following command adds GA fillers of type `GAFILL4` with regular spacing 20 and start location (0, 0).

  ```
  add_gate_array_filler -cell GAFILL4 -step_x 20 -step_y 20
  ```

- The following command adds GA fillers of type `GAFILL4` with vertical and horizontal spacing of 10 microns in the area defined by coordinates 100 100 500 500.

  ```
  add_gate_array_filler -area 100 100 500 500 -cell GAFILL4 -stepx 10 -stepy 10
  ```

Related Information

- `eco_design`
- `place_eco`
- `place Category Attributes`

add_mim_cap

```
add_mim_cap
[-help]
-cell cellName
[-orient instanceName]
[-prefix prefixName]
[{-step_x step_in_microns -step_y step_in_microns} [-area {x1 y1 x2 y2}]]
[{-area {x1 y1 x2 y2} | -location {x y}}] [-location {x y} | -offset {x y}]
```

Adds Metal-Insulator-Metal (MIM) cap into the design. This command is used before placing the standard cells.

Parameters

`-area x1 y1 x2 y2`	Specifies the area box, in microns, that will be used to insert Mim caps. If this option is not defined, the whole core area will be used. This option can be used only with -`step_x` and `-step_y` and is excluded with `-location`. *Data_type*: rect, required
`-cell` *cellName*	Specifies the name of the cell. *Data_type*: string, required
`-help`	Outputs a brief description that includes type and default information for each `add_mim_cap` parameter. For a detailed description of the command and all of its parameters, use the man command: `man add_mim_cap`.
`-location {x y}`	Inserts one Mim cap only at the given location. *Data_type*: point, required
`-offset {x y}`	Shifts the area used to insert mim caps with the specified offset value in both horizontal and vertical direction. This option can be used only with -`step_x` and `-step_y` and is excluded with `-location`. *Data_type*: point, required
`-orient` *orientation*	Specifies the orientation of the mim cap cell. *Default*: `r0` *Data_type*: string, optional
`-prefix` *prefixName*	Specifies the prefix for the inserted mim cap. *Data_type*: string, optional
`-step_x` *step_in_microns*	Specifies the distance between mim caps in the horizontal direction. This option must be defined with `-step_y` and is excluded with `-location`. *Data_type*: float, optional
`-step_y` *step_in_microns*	Specifies the distance between mim caps in the horizontal direction. This option must be defined with `-step_x` and is excluded with `-location`. *Data_type*: float, optional

Examples

- The following command inserts the mim cap cell `mim_cell1`:

```
add_mim_cap -cell mim_cell1 -orient r0 -step_x 34.56 -step_y 34.56 -offset { 9.317 8.336 }
```

Related Information

- `delete_mim_cap`

add_spare_insts

```
add_spare_insts
[-help]
[-async netName]
[-clock netName]
-file fileName
[-fix_placed_insts {TRUE|FALSE}]
[-hinst moduleName | -power_domain domainName [-spare_module spareModuleName]]
[-prefix prefix]
[-tie {0 | 1}]
```

Specifies a file that lists spare cells to add to the netlist. You can use this command before or after standard cell placement.

- If you use this command before placement, the `place_design` command distributes and places the added instances without including them in spare modules. It adds the instances to the top level of the design by default.
- If you use this command after placement, the software places the added instances without creating spare modules or disturbing the existing placement. The distribution of the cells is dependent upon the number of instantiations of each cell type.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `add_spare_insts` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man add_spare_insts`.
`-async netName`	Specifies which net(s) to tie off the async pin(s). Default: async pins will be tied to the logical zero net. *Data_type*: string, optional
`-clock netName`	Specifies the clock net that is tied to the clock pins. *Data_type*: string, optional

`-file` *`fileName`*	Specifies a text file that lists the spare cells, and the number of each, to add to the netlist. The file has the following format: "*`cellName number`*" Begin each line of a comment with a number sign (`#`). *Data_type*: string, required
`-fix_placed_insts {TRUE|FALSE}`	Specifies not to move the existing placed instances to find location for newly added spare instances. *Data_type*: bool, optional
`-hinst` *`moduleName`*	Specifies the name of the module to which spares will be added. *Data_type*: string, optional
`-power_domain` *`domainName`*	Specifies the name of the power domain to which spares will be added. *Data_type*: string, optional
`-prefix` *`prefix`*	Specifies a prefix for the names of the newly added cells. *Default:* `FE_spare` *Data_type*: string, optional
`-spare_module` *`spareModuleName`*	Specifies the name of the module created for spares. *Data_type*: string, optional
`-tie {0 | 1}`	Specifies the logical net to which the cells are connected. *Default:* `1` *Data_type*: int, optional

Example

The following command adds instances of the cells specified in `spare_file` and ties them to logical net `0`. It ties the clock pins to `sysclk`:

```
add_spare_insts -file spare_file -tie 0 -clock sysclk
```

`spare_file` contains the following:

```
DFFNSRX2T1T10    1
INVX4T1T10       2
NOR2X2T1T10      2
```

The following command adds instances of the cells specified in spare.txt.It ties the async pin RN to net VDD:

```
add_spare_insts -file spare.txt -async VDD:RN
```

`spare.txt` contains the following:

```
SDFFRHQXL 5
```

Related Information

- `gui_clear_spare_cells`
- `place_design`
- `reset_spare_insts`
- `set_spare_insts`
- Place Menu chapter in *Innovus Stylus Common UI Menu Reference.*

add_tap_walls

```
add_tap_walls
[-help]
[-bottom]
-cell
[-layer layer_name]
[-left]
[-orientation {r0 mx my r180}]
[-pitch distance]
[-prefix prefix]
[-range x1 y1|x2 y2]
[-right]
[-top]
[-y_flip_side {left right both}]
```

Adds tap walls to the assigned left, right, top, and/or bottom side of a block.

Parameters

`-bottom`	Specifies the add tap-wall cell on the bottom side. *Data_type*: bool, optional
`-cell cellName`	Specifies the name of the cell to insert. *Data_type*:
`-help`	Outputs a brief description that includes type and default information for each `add_tap_walls` parameter. For a detailed description of the command and all of its parameters, use the man command: `man add_tap_walls`.
`-left`	Specifies the add tap-wall cell on the left side. *Data_type*: bool, optional
`-layer layer_name`	Specifies the layer name to check rule. *Data_type*: string, optional

`-orientation {r0 mx my r180}`	Specifies orientation of the added tap-wall cells. *Data_type*: enum, optional
`-pitch` *distance*	Specifies the distance from the center of one tap-wall cell to the center of the following wall-tap cell. *Data_type*: float, optional
`-prefix` *prefix*	Specifies the prefix of tap-wall cells to inserted. *Data_type*: string, optional
`-range` *x1 y1|x2 y2*	Specifies the {x1 y1} or {x2 y2} as the start or end point at the user given side. *Data_type*: point, optional
`-right`	Specifies the add tap-wall cell on the right side. *Data_type*: bool, optional
`-top`	Specifies the add tap-wall cell on the top side. *Data_type*: bool, optional
`-y_flip_side {left right both}`	Specifies the orientation of the added tap-wall cells. Defines whether flipping tap walls located at certain side in the Y direction if site symmetry allows it. By default, both left and right side tap walls are R0 or MX. If -y_flip_side is set to left, the left side tap walls would be MY or R180. If it is set to both, both left and right side tap walls would be MY or R180. *Data_type*: enum, optional

add_well_taps

```
add_well_taps
[-help]
[-area  x1   y1   x2   y2 ]
[-avoid_abutment]
[-cell  cellName ]
[-cell_interval  microns]
[-channel_offset offset_value]
[-checker_board]
[-fixed_gap]
[-in_row_offset  microns ]
[-power_domain  powerDomainName ]
[-prefix  prefixName ]
[-site_offset  number_of_sites ]
[-skip_row  number ]
[-start_row  number ]
[-termination_cells  termination_cell_list | {-top_termination_cell cell_list -
bottom_termination_cell cell_list}}-column_cells  tap_column_cell_list} [-
block_boundary_only {true | false}]]
[-well_cut_cells  list_of_well_cut_cells ]
[-fix_well_cut_cells]
[-pitch microns [-pitch_offset microns]]
[{-check_channel  | -vertical_boundary_spacing spacing_with_well_tap | -incremental
list_of_cells}]
[{-insert_cells list_of_cells | -incremental list_of_cells}]
```

Adds physical-only well-tap cells. Well-tap cells are physical-only filler cells that are required by some technology libraries to limit resistance between power or ground connections and wells of the substrate. Well-tap cells are placed in a preplaced status.

- To delete well-tap cells, use the `delete_filler` command.
- To verify well-tap placement, generate markers for violations caused by well-tap cells, and create a violation report, use the `check_well_taps` command.

Parameters

`-area x1 y1 x2 y2`	Specifies the coordinates of the bounding box, in microns. By default, the core box that encloses all core rows is used. • `x1`: Specifies the area's lower left point on the x axis. • `y1`: Specifies the area's lower left point on the y axis. • `x2`: Specifies the area's upper right point on the x axis. • *y2:* Specifies the area's upper right point on the y axis. *Data_type*:
`-avoid_abutment`	Specifies whether inserting well-taps avoids abutment in vertical and horizontal direction. Well-taps on every other row are shifted by `-site_offset`. *Data_type*:
`-block_boundary_only {true | false}`	Specifies whether to insert termination tap cells only for well tap columns on the block boundary. Default is false, termination cells are inserted on both core boundary row and block boundary row. *Data_type*: **Note:** This parameter works only when inserting termination cells.
`-bottom_termination_cell cell_list`	Specifies the list of termination cells to be used at the bottom. *Data_type*:
`-cell cellName`	Specifies the name of the cell in the technology library to use as a well-tap. The cell should be of class type CORE. *Data_type*:
`-cell_interval microns`	Specifies the maximum distance from the center of one well-tap cell to the center of the following well-tap cell in the same row. If well-tap cells cannot be placed at this location due to a placement blockage, the software searches for a legal location: first, towards the previous well-tap cell, and then farther away. Note that the minimum cell interval is 0.45 times the cell interval.
`-channel_offset offset_value`	Specifies the offset value between taps and blockage in channel. *Minimum*: 0 *Data_type*:

`-check_channel`	Inserts well taps inside channel. *Data_type*:
`-checker_board`	Places the well-tap cells in a checkered pattern, that is, with every other row offset by one-half `cell_interval`. In a row immediately above or below a row where the pattern can't be maintained (for example, bottom and top row, or row immediately above or below macros) well-taps are inserted with a pitch of one-half `cell_interval` in the section where the pattern can't be maintained. When a floorplan has hard blocks or placement blockages, the rows above and below the block (or blockage) will also have well-taps at one-half gap. *Data_type*: **Note:** When this parameter is specified, `-skip_row` and `-avoid_abutment` are ignored.
`-column_cells tap_column_cell_list`	Specifies the list of well tap columns that need to be terminated. This parameter must be specified for termination tap cell insertion. *Data_type*:
`-fix_well_cut_cells`	Fixes all the instances which belong to the `wellCutCell` list. *Data_type*:
`-fixed_gap`	Ensures that the specified gap is always maintained. That is, the software does not search for a legal location if it cannot place well-tap cells at the specified `-cell_interval`. Instead, it goes on to place well-tap cells at the next position. *Data_type*: You can modify the gap by using the `-pitch` and `-pitch_offset` parameters.

`-help`	Outputs a brief description that includes type and default information for each `add_well_taps` parameter. For a detailed description of the command and all of its parameters, use the man command: `man add_well_taps`.
`-incremental` *`list_of_cells`*	Checks whether any instances of the cells specified in the *`list_of_cells`* are within 0.45 times the `-cellGap` from the next well-tap cell in the same row. If so, does not add well-tap cells at that location. *Data_type*:
`-insert_cells` *`list_of_cells`*	Inserts mixed well-taps, and supports the LUP layer and boundary layer rules. *Data_type*:
`-in_row_offset` *`microns`*	Specifies the offset, in microns, of the first well-tap cell from the start of the site row. *Default*: `0` (first cell on beginning of row is the starting point). *Data_type*:
`-pitch` *`microns`*	Used with the `-fixed_gap` parameter to give the software added flexibility to place well-tap cells if it is unable to place a cell at the specified location. If you specify this parameter, the software first tries to fit the well-tap cells in the locations specified by `-cell_interval` or forced by `-fixed_gap`; if it cannot do so, it adds well tap cells at the closest location allowed by this parameter. *Data_type*:
`-pitch_offset` *`microns`*	Used with the `-fixed_gap` and `-pitch` parameters to give the software greater flexibility. If you specify this parameter, the software first tries to fit the well-tap cells in the locations specified by `-cell_interval` or forced by `-fixed_gap`; if it cannot do so, it adds well tap cells at the closest location allowed by this parameter. *Data_type*:

`-power_domain` *`powerDomainName`*	Specifies the power domain in which the well-tap cells are inserted. This is the area where the well-tap cells are placed and connected to the correct power and ground nets. *Data_type*:
`-prefix` *`prefixName`*	Specifies the prefix for the placed well-tap cells. The software adds an underscore and a unique string to the prefix, in the following format: `prefix_`*`uniqueString`*. *Prefix Default:* `WELLTAP` *Data_type*: ⓘ Do not include an underscore when you specify a prefix.
-site_offset *number_of_sites*	Specifies an offset value, in sites, when `-avoid_abutment` is specified. *Default:* `0` *Data_type*:
`-skip_row` *`number`*	Specifies the number of rows to skip between two rows of an identical pattern. *Data_type*: **Note:** If you specify `-checker_board`, this parameter is ignored. *Default*: `0` (No rows are skipped, so well-tap cells are added in the same pattern to every row)
`-start_row` *`number`*	Specifies the row number starting from the from bottom or left side of the design on which the well-tap cells must be placed. *Default*: 1 (first row from the bottom is the starting point) *Data_type*:

`-termination_cells` *cell_list*	Specifies the list of cells that can be used as termination taps. `add_well_taps` selects the best fit termination cells from this list and inserts them at the top and bottom of well tap column segment. The existing non-termination well taps are deleted as needed. If the well tap columns have already been terminated with a valid termination cell, the existing termination cell are kept in the incremental run and no changes are made. *Data_type*: **Note:** It is recommended that well tap columns are added before termination cell insertion. **Note:** This parameter must be specified for termination tap cell insertion.
`-top_termination_cell` *cell_list*	Specifies the list of termination cells to be used at the top. *Data_type*:
`-vertical_boundary_spacing` *spacing_with_well_tap*	Specifies vertical boundary spacing with well tap. `add_well_taps` will recognize all vertical boundaries and keep center of tap cells to these boundaries equal to or less than the specified distance when inserting. *Minimum*: -1 *Maximum*: 1.79769313486232e+308 *Data_type*:
`-well_cut_cells` `list_of_well_cut_cells`	Specifies the list of cells with the well cuts. The `add_well_taps -incremental` flow treats cells specified in this list as placement blockage because they cut the row and fix the latch up violations within them. *Data_type*:

Examples

- The following command places instances of well-tap cell `xyz` on every row in the design. Instances in the same row are at least 22.50 µm but no more than 50 µm apart. Each instance name is prefixed with `welltap_`.

```
add_well_taps -cell xyz -prefix welltap -cell_interval 50
```

- The following commands place instances of well-tap cell `xyz` in a checkered pattern. The first instance in the odd row numbers (`1`, `3`, `5`, and so on) is placed at `0` (zero) offset. The first instance in even row numbers (2, 4, 6, and so on) is placed at 50 µm offset. Instances in the same row are at least 45 µm but no more than 100 µm apart. Instances on the odd rows are named `wtap_odd_1`, `wtap_odd_2`, and so on. Instances on the even rows are named `wtap_even_1`, `wtap_even_2`, and so on.

```
add_well_taps -cell xyz -prefix wtap_odd -cell_interval 100 -skip_row 1
add_well_taps -cell xyz -prefix wtap_even -cell_interval 100 -skip_row 1 \
    -start_row 2 -in_row_offset 50
```

- The following command places instances of well-tap cell `WTAP` on every row in the design. Instances in the same row are at least 21.68 µm but no more than 48.2 µm apart. Before placing the instances, the software checks for cells named `WTAP` and `PSO1`, and if it finds any within the minimum gap (that is, 21.68 µm from the nearest well-tap cell in that row) it does not place any instances in that area.

```
add_well_taps -cell WTAP -incremental "WTAP PSO1" -cell_interval 48.2
```

- In the following commands, cell TAP is specified as general well tap and TAP_TERMINATION is specified as termination well tap. TAP cells and tap columns are inserted first, followed by TAP_TERMINATION cells as termination cells at the top and bottom of each TAP column.

```
add_well_taps -cell TAP -cell_interval 28


add_well_taps -termination_cells TAP_TERMINATION -column_cells TAP
```

Related Information

- `delete_filler`
- `check_well_taps`
- "Substrate Biasing" in the "Low Power Design" chapter in the Innovus Stylus Common UI User Guide.

- "Adding Well-Tap Cells" in the "Placing the Design" chapter in the Innovus Stylus Common UI User Guide.

add_well_taps Category Attributes

The root attributes in the add_well_taps category are:

- add_well_taps_avoid_abutment
- add_well_taps_avoid_vertical_well_abutment
- add_well_taps_block_boundary_only
- add_well_taps_bottom_tap_cell
- add_well_taps_bottom_termination_cell
- add_well_taps_cell
- add_well_taps_channel_offset
- add_well_taps_check_channel
- add_well_taps_column_cells
- add_well_taps_create_rows
- add_well_taps_disable_check_zone_at_boundary
- add_well_taps_in_row_offset
- add_well_taps_insert_cells
- add_well_taps_rule
- add_well_taps_site_offset
- add_well_taps_tap_function_cells
- add_well_taps_tap_termination_alignment
- add_well_taps_termination_align
- add_well_taps_termination_cells
- add_well_taps_top_tap_cell
- add_well_taps_vertical_boundary_spacing
- add_well_taps_top_termination_cell
- add_well_taps_well_cut_cells

To set the value of any root attribute, use the `set_db` command:

```
set_db attribute_name value
```

To get the current value of any root attribute, use the `get_db` command:

```
get_db attribute_name
```

To get the current value of all attributes in the `add_well_taps` category, use the following command:

```
get_db -category add_well_taps
```

add_well_taps_avoid_abutment

`add_well_taps_avoid_abutment {true|false}`

Default: `false`

Data_type: bool, read/write

Specifies whether the inserted taps should avoid horizontal and vertical abutment.

add_well_taps_avoid_vertical_well_abutment

`add_well_taps_avoid_vertical_well_abutment {none ntap ptap all_tap}`

Default: `none`

Data_type: enum {none ntap ptap all_tap}

Specifies the type of well-tap cells to be forbidden from the vertical abutment.

add_well_taps_block_boundary_only

`add_well_taps_block_boundary_only {true|false}`

Default: `false`

Data_type: bool, read/write

Specifies whether to insert termination tap cells only for well tap columns on the block boundary. Default is false, termination cells are inserted on both core boundary row and block boundary row.

Note: This attribute works only when inserting termination cells.

add_well_taps_bottom_tap_cell

`add_well_taps_bottom_tap_cell cellName`

Default: `""`

Data_type: string, read/write

Specifies the cell to be used as the bottom tap for the add_endcaps boundary cell insertion flow.

add_well_taps_bottom_termination_cell

add_well_taps_bottom_termination_cell *cellList*

Default: `""`

Data_type: string, read/write

Specifies the list of bottom termination cells to be used.

add_well_taps_cell

`add_well_taps_cell` *cellName*

Default: `""`

Data_type: string, read/write

Specifies the name of the cell in the technology library to use as a well-tap. The cell should be of class type CORE.

add_well_taps_channel_offset

`add_well_taps_channel_offset` *offset_value*

Default: `0.0`

Data_type: double, read/write

Specifies the offset value between taps and the blockage in channel.

add_well_taps_check_channel

`add_well_taps_check_channel {true|false}`

Default: `false`

Data_type: bool, read/write

Specifies whether the blockage channel will be checked.

add_well_taps_column_cells

`add_well_taps_column_cells` *`cellist`*

Default: ""

Data_type: string, read/write

Specifies the maximum distance, in microns, between the tie-cell driver and the tie-pins. A value of `0` implies no distance constraint.

add_well_taps_create_rows

`add_well_taps_create_rows {true|false}`

Default: `false`

Data_type: bool, read/write

Creates rows for welltap cell techSites which do not have rows in the floorplan.

add_well_taps_disable_check_zone_at_boundary

`add_well_taps_disable_check_zone_at_boundary {none vdd vss both}`

Default: `none`

Data_type: enum {none vdd vss both}, read/write

Creates rows for welltap cell techSites which do not have rows in the floorplan.

add_well_taps_in_row_offset

`add_well_taps_in_row_offset` *`offset_value`*

Default: `0.0`

Data_type: double, read/write

Specifies the distance between the first welltap in a row and the start of the row.

add_well_taps_insert_cells

`add_well_taps_insert_cells {`*`cellName rule dense_layer dense__rule boundary_layer boundary_rule`*`}`

Default: `""`

Data_type: string, read/write

Inserts well-taps based on cell, layer, and interval.

- *`cellName`*: Specifies the cell name.
- *`rule`*: Cell with this rule would be added as a general tap cell for all placeable area. The center to center spacing of tap cells would be `≤ (2*$rule)` without `-checker_board` and `≤ (4*$rule)` with -`checker_board`.
  Note: Center of the tap cell is used to check the spacing.
- *`dense_layername`*: Specifies the dense layer name.
- *`dense_rule`*: Cell with dense layer would be added to the given layer regions only. Cell with dense layer must follow the dense rule. The center to center spacing of dense tap cells would be `≤ (2*$dense_rule)` without `-checker_board` and `≤ (4 *$dense_rule)` with `-checker_board`.
- *`boundary_layername`*: Specifies the boundary layer name.
- *`boundary_rule`*: A cell would be added for boundaries inside the given layer regions. The spacing between the boundary to the center of boundary tap cells would be ≤ $ boundary rule. Boundary layer regions which cover partial sites should be extended. Boundary taps can be located in or out of boundary layer regions.

Note:

- You can define multiple cells for -insert_cells by separating each cell by { }.
- There should be at lest one rule, dense layer or boundary layer, defined for each cell.
- The order of tap cell insertion is:
  a. Dense tap
  b. Boundary tap
  c. General tap
- If the area is covered by both dense layer region and boundary layer region, both layer rules must be satisfied.
- If `-insert_cells` is defined multiple times, the last setting is accepted by the tool.

add_well_taps_rule

`add_well_taps_rule` *`RuleValue`*

Default: `0.0`

Data_type: double, read/write

Specifies the rule to be honored when adding well taps.

add_well_taps_site_offset

`add_well_taps_site_offset` *`number_of_sites`*

Default: `1`

Data_type: int, read/write

Specifies an offset value, in number of sites, so that the vertical taps are aware of.

add_well_taps_tap_function_cells

`add_well_taps_tap_function_cells` *`cell_list`*

Default: `""`

Data_type: string, read/write

Specifies the list of tap function cells to be used.

add_well_taps_tap_termination_alignment

`add_well_taps_tap_termination_alignment [true | false]`

Default: `false`

Data_type: bool, read/write

Specifies whether to align well-taps and termination cells.

add_well_taps_termination_align

`add_well_taps_termination_align {left right center}`

Default: `left`

Data_type: enum {left right center}, read/write

Specifies the termination cell alignment.

add_well_taps_termination_cells

`add_well_taps_termination_cells` *`cellist`*

Default: `""`

Data_type: string, read/write

Specifies the list of cells that can be used as termination taps. The command `add_well_taps` selects the best fit termination cells from this list and inserts them at the top and bottom of well tap column segment. The existing non-termination well taps are deleted as needed. If the well tap columns have already been terminated with a valid termination cell, the existing termination cell are kept in the incremental run and no changes are made.

Note: It is recommended that well tap columns are added before termination cell insertion.

Note: This attribute must be specified for termination tap cell insertion.

Related Commands

- `add_well_taps`

add_well_taps_top_tap_cell

`add_well_taps_top_tap_cell` *`cellName`*

Default: `""`

Data_type: string, read/write

Specifies well taps inserted into top rows during enclosed end cap insertion.

Note: This attribute must be specified when `add_endcaps_boundary_tap` is set to `true`.

add_well_taps_vertical_boundary_spacing

`add_well_taps_vertical_boundary_spacing` *`spacing_value`*

Default: `-1.0`

Data_type: double, read/write

Specifies the spacing between the center of tap cell and vertical boundary.

add_well_taps_top_termination_cell

`add_well_taps_top_termination_cell` *`cellList`*

Default: `""`

Data_type: string, read/write

Specifies the list of top termination cells to be used.

add_well_taps_well_cut_cells

`add_well_taps_well_cut_cells` *`cellist`*

Default: `""`

Data_type: string, read/write

Specifies the list of cells with the well cuts. `addWellTap` considers the specified cells cutting the row.

check_filler

```
check_filler
[-help]
[-area x1 y1 x2 y2]
[-power_domain power_domain_name]
[-horizontal_max_length]
[-out_file fileName ]
[-report_gap microns | -adjacent_filler1]
[-vertical_stack_max_length]
```

Checks for locations that are missing filler cells, after adding the cells with the `add_fillers` command. Optionally, generates a report with details on sites that are missing filler cells.

If you run this command without any parameters, it reports the total number of sites in the core area that are missing filler cells.

Parameters

`-adjacent_filler1`	Checks for locations that are filled with adjacent one-site-width fillers. With this parameter, `check_filler` flags any one-site-width filler placed next to each other as a violation. To enable this parameter, you must first specify a one-site-width filler by using the following command: `set_db add_fillers_cells fillerCellList` *Data_type*: bool, optional
`-area {x1 y1 x2 y2}`	Specifies a bounding box for the area in which to check for missing filler cells. *Default:* Entire core area *Data_type*: rect, optional
`-horizontal_max_length`	Checks if the horizontal stack length violates the maximum limit. *Data_type*: bool, optional
`-out_file fileName`	Writes the output of the `check_filler` command to the specified file. *Data_type*: string, optional

`-help`	Outputs a brief description that includes type and default information for each `check_filler` parameter. For a detailed description of the command and all of its parameters, use the man command: `man check_filler`.
`-power_domain` *`power_domain_name`*	Checks for missing filler cells in the specified power domain. The software checks for missing filler cells in all power domains if you do not specify this parameter. *Data_type*: string, optional
`-report_gap` *`microns`*	Checks only for gaps of the specified size- ignores any other gap, whether larger or smaller. *Data_type*: float, optional
`-vertical_stack_max_length`	Checks if the vertical stack length violates the maximum limit. *Data_type*: bool, optional

Examples

The following command checks the entire core area after running `add_fillers` and reports the number of sites that are missing filler cells:

```
check_filler
```

The software reports the following:

```
*INFO: Total number of gaps found: 85
```

The following command checks within the specified bounding box at coordinates {0 0 100 100} and reports the number of sites that have a gap of exactly 0.14 microns:

```
check_filler -report_gap 0.14 -area {0 0 100 100}
```

The software reports the following:

```
*INFO: Total number of 0.140 microns width gap found: 3
```

Related Information

- add_fillers
- add_fillers Category Attributes
- delete_filler

check_library

```
check_library
[-help]
[-place [-cell cell_name|-all_lib_cell ][-legal_frequency ] [-file fileName] ]
```

Checks the problematic cells that cause issues such as bad geometries for DRC and bad pin access for routing. You can specify a library cell or a partition cell. It also performs checks for the multi-height cell legal frequency. The report is dumped into an output file.

Parameters

`-all_lib_cell`	Specifies to check all library cells. *Data_type*: bool, optional
`-cell cell_name`	Specifies the cell to check. *Data_type*: (base_cell)+, optional
`-file fileName`	Specifies the name of the output file. *Data_type*: string, optional
`-help`	Outputs a brief description that includes type and default information for each `check_library` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man check_library`.
`-legal_frequency`	Specifies to check multi-height cell legal frequency. *Data_type*: bool, optional
`-place`	Specifies to do library checking for placement. *Data_type*: bool, optional

Example

- The following command checks the `buf01` cell:

```
check_library -place -cell buf01

Estimated cell power/ground rail width = 0.021 um

*info: running library checker ...

*info: total 1 cell checked.

*reporting cell (buf01) ...

reporting cell drc : "legal"

reporting cell pin access : "legal"

reporting cell symmetry : cell (buf01) has lef symmetry X Y

reporting cell pin alignment : "legal"

reporting cell legal frequency ...
cell (buf01) legalization frequency: one legal location out of every 1.25 sites
```

check_place

```
check_place
[-no_check_pin_access]
[-clear_marker]
[-honor_preroute_for_filler]
[-ignore_filler_in_density]
[-ignore_out_of_core]
[-inst list_of_instances]
[-io_pin_blockage]
[-macro_blockage]
[-no_hard_fence]
[-no_preplaced]
[-no_halo]
[violationReportFileName]
[-selected_only]
```

Checks `FIXED` and `PLACED` cells for violations, adds violation markers to the design display area, and generates a violation report. Use the Violation Browser to see the violation markers.

Rules as to what constitutes a placement violation are different for `FIXED` and `PLACED` cells. For example, `FIXED` cells in obstruction areas are not considered to be violations, but `PLACED` cells are considered to be violations.

This command considers preplaced instances and macros as `FIXED`.

Note: The utilization value reported by `check_place` and the density value reported by `time_design` might differ because `check_place` considers padding when calculating utilization but `time_design` does not consider padding when calculating density.

You can delete padding by running the `delete_all_cell_padding` command before running `check_place`.

Note: If `set_db place_detail_legalization_inst_gap` is defined, check_place will report the gaps that violate the specified spacing.

The `check_place` command checks for the following violations:

- Implant max cover length. If this violation is detected, GUI adds markers for every instance. As per this violation rule the implant shapes cannot be fully covered horizontally by neighbor shapes on both top and bottom opposite sides. If check_place detects the violation, GUI adds markers for every instance
- Region and fence violations (Instances placed outside their region or fence)
- Out of core area violations (Instances placed partly or wholly outside the core area)

- Not placed on placement grid (Standard cells not on the placement grid, or blocks not on the manufacturing grid). If the design is using the FinFET technology, check_place will also check if standard cells and blocks are not aligned on the FinFET grid and report violations.
- Row orientation violations (Instances whose orientation is illegal for their row; may short power to ground)
- Tech site violations (Instances placed on tech sites not matching their cells and instances that do not have rows for their tech sites)
- Overlapping with other instance (Placed instances that overlap with other instances)
- Pin access violations (Instances with at least one signal pin that does not have routing access due to pre-wires)
- Orientation violations (Instances whose orientation is not legal based on its cell symmetry rules)
- Routing/placement blockage violations in the core area (Instances placed in an obstruction created by pre-wires or routing blockage)
  The command does not check fixed macros.
- Placement blockage violations in the core area (for example, placed instances in an obstruction created by prerouted wires or a routing blockage, placed instances that overlap a placement blockage area).
  Note: The command does not check fixed instances overlapping placement or routing blockages by default.
- Site orientation violations (Instances whose orientation does not match the tech site orientation)
- Not of fence violations (Instances placed inside a fence to which it does not belong)
- Prints implant layer names in the Implant Layer Spacing violation. The implant Spacing Violation would be as follows:

  ```
  Implant_Spacing_Violation
  $instance1 implant layer $layer1 has corner touch violation to $instance2 implant
  layer $layer2
  VT group name: $Group  Base layer: $layer
  MinWidth: $width  MinSpacing:$spacing    ShiftDistance: $value
  Box=$box
  ```

  Note: It only works for the static check of the `checkPlace` command.

In addition to checking for violations, the command reports the following:

- Number of placed instances in the design
- Number of unplaced macros in the design

- Placement density, in terms of used sites over available sites

Parameters

`-no_check_pin_access`	Specifies whether to check for pin access violations. *Data_type*: bool, optional
`-clear_marker`	Removes the violation markers from a previous `check_place` run, without rerunning `check_place`. *Data_type*: bool, optional
`-honor_preroute_for_filler`	Checks for placement blockage violations on physical instances placed under preroutes or specified by the `place_detail_preroute_as_obs` attribute. By default, `check_place` does not treat filler cells placed under preroutes as violations, even though they are placement/routing blockages; however, it does treat standard cells placed under preroutes as violations. *Data_type*: bool, optional
`-ignore_filler_in_density`	Ignores filler cells in calculating utilization. *Data_type*: bool, optional
`-ignore_out_of_core`	Skips instances outside the core design area. *Data_type*: bool, optional
`-inst` *`list_of_instances`*	Specifies the list of instances to check. *Data_type*: string, optional
`-io_pin_blockage`	Instructs legalization routines to treat top-level I/O pins as fixed metal obstructions while placing standard cell instances at the block level. There is a minor run-time penalty with this parameter, so it is not turned on by default. *Data_type*: bool, optional

`-macro_blockage`	Checks for overlap between hard macro and macro-only blockage and reports macro blockage violation. By default, macro-only blockage is ignored for both hard macro and standard cell. With this setting, it is treated as a placement obstruction for macros. *Data_type*: bool, optional
`-no_halo`	Skips block halos. Uses the place-and-route boundary for the placement overlap check. *Data_type*: bool, optional
`-no_hard_fence`	Ignores fence and region placement constraints. *Data_type*: bool, optional
`-no_preplaced`	Does not check fixed instances. When this parameter is specified, `check_place` treats fixed instances as placement blockages. *Data_type*: bool, optional
`-selected_only`	Specifies to check only the selected instances legality. *Data_type*: bool, optional
violationReportFileName	Specifies the report file. If you do not specify this parameter, the command displays a summary report. The report file is more detailed than the summary report, and lists every instance with a violation. *Data_type*: string, optional

Note: If a design is loaded where a macro is fixed in an orientation that is not supported according to its LEF, running `check_place` changes the orientation and the position of the macro, even if the `-no_preplaced` option is specified.

Example

Here is an example:

Cell positions before check_place:

```
FPFEL0_SLM/fpicl/fpiclmsc/ThermRemote_FP_1: {544.32 648}
FPFEL0_SLM/fpicl/fpiclmsc/ThermRemote_FP_2: {544.32 669.6}

< CMD> check_place
**WARN: (ENCDB-1266): Both given orientation and original orientation not legal,
```

```
FPFEL0_SLM/fpicl/fpiclmsc/ThermRemote_FP_1 changes to orientation R0
**WARN: (ENCDB-1266): Both given orientation and original orientation not legal,
FPFEL0_SLM/fpicl/fpiclmsc/ThermRemote_FP_2 changes to orientation R0
Design has 3 cells with cell padding or block halo.
Begin checking placement ... (start mem=690.9M, init mem=714.5M)
*info: Placed = 1098
*info: Unplaced = 0
Placement Density:0.00%(0/407468)
Finished check_place (cpu: total=0:00:07.6, vio checks=0:00:00.0; mem=696.2M)
```

Cell positions after check_place:

```
FPFEL0_SLM/fpicl/fpiclmsc/ThermRemote_FP_1: {544.446 648}
FPFEL0_SLM/fpicl/fpiclmsc/ThermRemote_FP_2: {544.446 669.6}
```

Related Information

- check_drc
- delete_all_cell_padding
- place Category Attributes
- time_design
- Tools Menu chapter in the *Innovus Stylus Common UI Menu Reference*.
    - Violation Browser

create_gate_array_filler_group

```
create_gate_array_filler_group
[-help]
-gate_array_cells gaCellList
-gate_array_fillers gaFillerCellList
-group groupName
```

It creates gate array filler groups for the post-mask `place_eco` flow. When a filler group is created, `place_eco -use_gate_array_filler_groups true` can only place the GA cells at the GA filler location that is included in the same filler group.

This command can be used multiple times to define several gate array groups for `place_eco`. If one filler group name is defined many times, the last one will be kept.

Parameters

`-gate_array_cells` *gaCellList*	Specifies the list of GA cells. The * wildcard is supported to specify the cell list. *Data_type*: string, required
`-gate_array_fillers` *gaFillerCellList*	Specifies the list of GA filler cells. **Note**: One GA cell or GA filler cell cannot be defined in two groups, otherwise it will give error. *Data_type*: string, required
`-group` *groupName*	Specifies the group name. *Data_type*: string, required

Example

The following commands show the flow for GA Filler groups:

```
innovus> create_gate_array_filler_group -gate_array_cells GA_cell_list -
gate_array_fillers GA_filler_list -group group

innovus> place_eco -use_gate_array_filler_groups true
```

Related Information

- delete_gate_array_filler_group
- place Category Attributes
- place_eco

create_inst_space_group

```
create_inst_space_group
[-help]
-group_name groupName
-inst listOfInstances
{[{[ -spacing_y Distance] [-spacing_x Distance]} [-checking_box {all|cross_only}]] | [-
spacing_radial Distance]}
```

Creates space group for instances with a specific vertical or horizontal-distance constraint. You need to run the following command before `create_inst_space_group`.

```
>set_db place_detail_check_inst_space_group true
```

The `create_inst_space_group` information can be saved in DB.

Parameters

`-checking_box {all cross_only}`	Checks all area or the intersection area from `-spacing_x` & `-spacing_y`. • `cross_only`: Checks the intersection area of -`spacing_x` and -`spacing_y`. If any instance violates both -`spacing_x`and -`spacing_y`, it is marked as a violation. • `all`: Checks all area covered by `spacing_x` and `spacing_y`. If any instance violates -`spacing_x` or -`spacing_y`, it is marked as a violation. The constrained box is ended by core/blockage/power domain/hard macro/row boundary. *Default*: `cross_only` *Data_type*: enum, optional
`-group_name` *groupName*	Specifies the name of the instance group. *Data_type*: string, required
`-help`	Outputs a brief description that includes type and default information for each `create_inst_space_group` parameter. For a detailed description of the command and all of its parameters, use the man command: `man create_inst_space_group`.
`-inst` *listOfInstances*	Specifies all instances that belong to the same space group by name. *Data_type*: string, required
`-spacing_radial` *Distance*	Specifies the radial distance, in microns. *Data_type*: float, optional
`-spacing_x` *Distance*	Specifies the distance, in microns, between each specific instance in the X direction. *Data_type*: float, optional
`-spacing_y` *Distance*	Specifies the distance, in microns, between each specific instance in the Y direction. *Data_type*: float, optional

Related Information

- delete_inst_space_group

create_pg_keep_out

```
create_pg_keep_out
[-help]
[-bottom value]
[-left value]
[-pin <string>]
[-right value]
[-top value]
{-cell lib_cell_name | -inst instance_name} {-layer layer_name | -layer_id  layer_id}
```

Defines the PG keepout box for a cell or a pin.

Parameters

`-bottom value`	Specifies the value to expand the keepout box on the bottom side. *Data_type*: float, optional
`-cell` *`lib_cell_name`*	Specifies the lib cell that needs to be constrained by the PG keepout box. *Data_type*: string, required
`-help`	Outputs a brief description that includes type and default information for each `create_pg_keep_out` parameter. For a detailed description of the command and all of its parameters, use the man command: `man create_pg_keep_out`
`-inst` *`instance_name`*	Specifies the instance that needs to be constrained by the PG keepout box. *Data_type*: string, required
`-layer` *`layer_name`*	Specifies the layer on which the PG keepout box is applied. The layer is the LEF/OA name of routing layers. *Data_type*: string, required
`-layer_id` *`layer_id`*	Specifies the layer ids on which the PG keepout box is applied. The layer id is the layer number of routing layers. *Data_type*: string, required
`-left` *value*	Specifies the value to expand the keepout box on the left side. *Data_type*: float, optional
`-pin` *`pin_name`*	Specifies the pin that needs to be constrained by the PG keepout box. *Data_type*: string, optional
`-right` *`value`*	Specifies the value to expand the keepout box on the right side. *Data_type*: float, optional
`-top` *`value`*	Specifies the value to expand the keepout box at the top. *Data_type*: float, optional

Example

The following command keeps `mycustomcell/the_m4_pin` away from M5 power stripes:

```
create_pg_keep_out -cell mycustomcell -pin the_m4_pin -layer M5
```

Related Information

- `delete_pg_keep_out`

create_pipeline_bus_guide

```
create_pipeline_bus_guide
[-help]
[-net_group netGroupNames]
[-width_factor value]
[-layer id1:id2]
[-report fileName]
[-spacing value {-report filename}]
```

Generates bus-guide for the pipeline netgroup. The bus-guide is used to guide the router for expected path. It can be used in the analysis flow or after the pipeline flops are inserted in the netlist.

Note: The bus-guide width is scalable by the width factor. Default width factor is 1.0. The bus width equals to the (width factor) x (the largest pitch of given routing layers) x (the number of bus bits) + 1 extra pitch. The extra pitch is added to account for the possibility that bus-guides do not line up exactly with the routing grid.

Parameters

`-net_group` *netGroupNames*	Specifies the netgroup name(s) that the bus-guide is created for. *Default*: `""` *Data_type*: string, optional
`-width_factor` *value*	Specifies the width factor of bus-guide. *Default*: `1.0` *Data_type*:
`-layer` *id1:id2*	Specifies the layer(s) to use for the generated bus-guide. It can be a single layer or layer range. *Default*: `all` *Data_type*: float, optional
`-report` *fileName*	Specifies the analysis report file name. *Default*: `<filename>.rpt` *Data_type*: string, optional
`-spacing` *float*	Specifies the maximum space between flops. *Default*: `""` *Data_type*: float, optional

Example

- The following command writes out a report file 'busguide.rpt' which tells the total estimated length of bus (in microns), the number of stages that will be needed (based on the given spacing of 200 um), and location of each stage of PFFs for all net groups on all routing layers.
  ```
  create_pipeline_bus_guide -spacing 200 -report busguide.rpt
  ```
- The following command create bus-guide for net group 'NG1' on routing layers 2, 3, and 4. The bus-guide width is 2 times the default width.
  ```
  create_pipeline_bus_guide -net_group NG1 -width_factor 2.0 -layer 2:4
  ```

create_pipeline_net_group

```
create_pipeline_net_group
[-help]
-net_group_name netGroupName
[-end_reserve reserveDistanceInMicrons]
[-end_term termList]
[-inst instNameListOrRegularExp]
[-net listOfNetnamesOrRegularExp]
[-start_reserve reserveDistanceInMicrons]
[-start_term termList]
```

Creates Net Groups for Pipeline Flip-Flop (PFF) Placement and Reporting. This command can be used for analysis before the flops exist in the netlist; or it can be used post flop insertion to add a new net group.

Parameters

`-end_reserve` *reserveDistanceInMicrons*	Specifies a distance value from the end points on block boundary to the internal connection inside the block, which is accounted for in each stage distance value. The unit is in micron. *Data_type*: float, optional
`-end_term` *termList*	Specifies the end points of net groups. *Data_type*: string, optional
`-net_group_name` *netGroupName*	Specifies the name of the net group. *Data_type*: string, required
`-inst` *instNameListOrRegularExp*	Creates net group of all nets connected to the given instances. All the instances must be flip-flops or latches. All the nets attached to the D and Q pins are put into the PFF net group. *Data_type*: string, optional
`-net` *listOfNetNamesOrRegularExp*	Creates net group of all given nets. *Data_type*: string, optional
`-start_reserve` *reserveDistanceInMicrons*	Specifies a distance budget from internal connections inside block to the start point of the PFF net. This value is added to the first stage distance when calculating equidistant spacing for each stage. The unit is in micron. *Data_type*: float, optional
`-start_term` *termList*	Creates net group of all data nets between the given end points. The PFF-chain is traced from the startTerm, through D and Q terminals of PFF(s) until an endTerm is matched. *Data_type*: string, optional

Example

- The following command creates a net group named `netGroup1' for Pipeline Flip-Flop (PFF) of all data nets between `blockA/DATA_OUT*` and `blockB/DATA_IN*`:

  ```
  create_pipeline_net_group netGroup1 -start_term blockA/DATA_OUT* -end_term
  blockB/DATA_IN*
  ```

- The following command creates a net group named `netGroup2` for all nets connected with all instance names that begin with `PIPELINE_INST/in`:

  ```
  create_pipeline_net_group NetGroup2 -inst PIPELINE_INST/in*
  ```

- The following command creates a net group named `netGroup3` for all net names that begin with `SUM_P4`.

  ```
  create_pipeline_net_group NetGroup3 -net SUM_P4*
  ```

Related Information

- delete_pipeline_net_group
- update_pipeline_net_group

create_scan_chain

```
create_scan_chain
[-help]
[-name]
[-start {ftname | instPinName}]
[-stop {ftname | instPinName}]
```

Specifies a scan chain or a group in a design. The actual tracing of the scan chain is performed by the `trace_scan` or `reorder_scan` command. If scan chain specification fails, use the `trace_scan -verbose` command to see where the scan chain stops.

Note: Scan groups are sometimes called scan partitions.

ⓘ This command is not necessary for scan chains that are already specified in DEF or TDF files.

Parameters

-name	Specifies the scan chain or scan group in a design. *Data_type*: string, required
-start {*ftname* | *instPinName*}	Specifies the: • Starting scan pin name for the scan chain or scan group. • Design input pin name. • linstance output pin name. *Data_type*: string, required
-stop {*ftname* | *instPinName*}	Specifies the: • Stopping scan pin name for the scan chain or scan group. • Design output pin name. • Instance input pin name. *Data_type*: string, required

Example

The following command sets scan chain `test_si` with a starting I/O pin name and a stopping pin name:

```
create_scan_chain test_si -start test23 -stop test_so
```

Related Information

- `delete_scan_chain`
- `reorder_scan`
- `report_scan_chain`
- `trace_scan`
- Placing the Design chapter in the *Innovus Stylus Common UI User Guide*
  - Optimizing and Reordering Scan Chains

create_scan_chain_partition

```
create_scan_chain_partition
-name partitionName
{-all | -scan_chains chainName}
```

Defines a group (a partition) of compatible scan chains for the scan reordering flow. The software can swap elements in scan chains in the partition. Swapping elements helps reduce the total scan wire length in the partition and improves local congestion in the design.

To enable swapping, type the following command:

```
set_db reorder_scan_allow_swapping true
```

Parameters

`-all`	All scan chains in the specified partition are compatible. *Data_type*: bool, required
`-scan_chains chainName`	Only the specified scan chains in the specified partition are compatible. *Data_type*: string, required
`-name partitionName`	Specifies a name for the partition and a list of the scan chains that belong to the partition. The scan flops in these chains are swappable. *Data_type*: string, required

Example

The following commands define two scan chains using `create_scan_chain` command, and define one partition using `create_scan_chain_partition`. The `reorder_scan_allow_swapping` attribute allows the software to swap elements between compatible chains.

```
create_scan_chain chain1 -start SI1 -stop SO1

create_scan_chain chain2 -start SI2 -stop SO2

create_scan_chain_partition -name clk1Domain chain1 chain2

set_db reorder_scan_allow_swapping true

reorder_scan
```

Related Information

- `create_scan_chain`
- `delete_scan_chain_partition`
- `report_scan_chain_partitions`
- `reorder_scan Category Attributes`
- Placing the Design chapter in the *Innovus Stylus Common UI User Guide*
  - Optimizing and Reordering Scan Chains

create_spare_module

```
create_spare_module
-module_name moduleName
-cells {cellName [number] [cellName [number] ] …}
[-clock netName]
[-reset netName:pinName [pinName] …]
[-tieoffs {tieCellName [tieCellname] } ]
[-tie_low {pinList | *} ]
[-use_cell_as_prefix]
```

Creates a spare module made up of the specified cells.

Parameters

`-cells {`*`cellName`* `[`*`number`*`] [`*`cellName`* `[`*`number`*`] ] …}`	Specifies the names of the cells, and the number of each, to add to the spare module. If you do not specify the number of cells to add, the `set_db add_tieoffs_max_fanout` attribute controls the number of cells. *Data_type*: string, required
`-clock` *`netName`*	Specifies a clock net to connect to the clock pins of the instances in the spare module. *Data_type*: string, optional
`-module_name` *`moduleName`*	Specifies a name for the spare module. *Data_type*: string, required
`-reset` *`netName`*`:`*`pinName`* `[`*`pinName`*`]`	Specifies a net and the pins to connect to the reset pins of sequential spare cells in the spare module. If you do not specify this parameter, these reset pins are not connected. *Data_type*: string, optional
`-tieoffs` *`tieCellName`* `[`*`tieCellName`*`]`	Specifies tie-high and tie-low cells to add to the spare module. The software recognizes whether the pins are high or low. *Data_type*: string, optional
`-tie_low {`*`pinList`* `| *}`	Specifies the pins to connect to the specified tie-low cell. Pins not specified here will be connected to `1'b1` or the tie-high cell. If no tie-low cells are specified with the `-tieoffs` parameter, it connects the pins to `1'b0`. *Data_type*: string, optional Use the following format for the pin list: `<pinList>` :: {*`cellName`*: <*`pin`*>} … `<pin>` :: *`pinName`* |*
`-use_cell_as_prefix`	Adds the master cell name as a prefix to the name of gates that are instantiated inside the module. The software adds `spr_gate_`*`n`* as a prefix by default; when you specify `-use_cell_as_prefix`, it prepends the master cell name to the default prefix. *Data_type*: bool, optional

Examples

The following command creates spare module `mod1` with cells `cellA`, `cellB`, and `tiehi4`:

```
create_spare_module -cells {cellA cellB} -tieoffs tiehi4 -module_name mod1
```

The following command creates spare module `mod1` with the following gates: `and_spr_gate_1`, `and_spr_gate_2`, `nand_spr_gate_1`, `nor_spr_gate_1`:

```
create_spare_module -cells {and 2 nand nor} -module_name mod1 -use_cell_as_prefix
```

Related Information

- `add_tieoffs Category Attributes`
- `delete_spare_module`

delete_all_cell_padding

`delete_all_cell_padding`

Removes placement padding specified by `set_cell_padding` for all cells. If you run this command before running `check_place`, `check_place` does not consider the cell padding when calculating effective utilization. For more information, see `check_place` and `set_cell_padding`.

Parameters

None

Examples

The following set of commands specifies the cell padding for `HDSDFRSPQ1` and `HDSDFRSPQ2`, and then deletes all padding by using `delete_all_cell_padding`:

```
set_cell_padding HDSDFRSPQ1 2
set_cell_padding HDSDFRSPQ2 2
delete_all_cell_padding
```

Related Information

- `check_place`
- `set_cell_padding`
- `report_cell_padding`
- Placing the Design chapter in the *Innovus Stylus Common UI User Guide*.
    - Adding Padding

delete_all_scan_cells

```
delete_all_scan_cells
[-help]
```

Unassigns all assigned scan cells. This command can be called after importing the design if there are scan cell specifications in library; or else it can be used after the `set_scan_cell` command.

Parameters

None

Example

The following set of commands deletes all scan cells after loading design:

```
read_db init.enc.dat top
delete_all_scan_cells
```

Related Information

- `read_db`
- `set_scan_cell`
- Placing the Design chapter in the *Innovus Stylus Common UI User Guide*.
    - Optimizing and Reordering Scan Chains

delete_base_cell_edge_type

```
delete_base_cell_edge_type
[-help]
-base_cell cellName
-edge_type edgeType
```

Deletes the cell edge type constraint of the specified cell.

Parameters

`-base_cell` *`cellName`*	Specifies cell name to remove constraint. *Data_type*: string, required
`-edge_type` *`edgeType`*	Specifies the name of edge type to be removed. *Data_type*: string, required
`-help`	Outputs a brief description that includes type and default information for each `delete_base_cell_edge_type` parameter. For a detailed description of the command and all of its parameters, use the man command: `man delete_base_cell_edge_type`.

Related Information

- `place Category Attributes`
- `report_cell_edge_type`

delete_cell_edge_spacing

```
delete_cell_edge_spacing
[-help]
type1
type2
```

Deletes the spacing rule between two edge types. The edge types specified must be predefined.

Note: This command only deletes the cell edge constraint specified by Tcl and does not delete the edge spacing defined in LEF.

Parameters

type1	Specifies edge type1 for edge spacing. *Data_type*: string, required
type2	Specifies edge type2 for edge spacing. *Data_type*: string, required
-help	Outputs a brief description that includes type and default information for each `delete_cell_edge_spacing` parameter. For a detailed description of the command and all of its parameters, use the man command: `man delete_cell_edge_spacing`.

Related Information

- `place Category Attributes`
- `report_cell_edge_spacing`

delete_cell_padding

```
delete_cell_padding
[-help]
base_cell
```

Deletes the padding for a cell or list of cells. If a regular expression or a list of cells is provided, this command returns the number of cells for which the padding was deleted.

Parameters

base_cell	Specifies the cell name for which padding has to be deleted. The cell name can be a regular expression, or a list of cell names or expressions. *Data_type*: string, required
-help	Outputs a brief description that includes type and default information for each `delete_cell_padding` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man delete_cell_padding`.

Examples

- The following command deletes padding for the cells with prefix `FF_SCAN`:
  ```
  delete_cell_padding FF_SCAN*
  ```
- The following command deletes padding for all cells that have names starting with `BUF`, `INV`, and `DLY`. If a matched cell name does not have padding information, it is ignored without any message.
  ```
  delete_cell_padding {BUF* INV* DLY*}
  ```
- The following command deletes padding for cell `FF_SCAN1` and cell `FF_SCAN2`:
  ```
  delete_cell_padding {FF_SCAN1 FF_SCAN2}
  ```

Related Information

- place Category Attributes
- report_cell_padding
- set_cell_padding

delete_cell_stack_area

```
delete_cell_stack_area
[-help]
[-cell cell_name]
[-group group_name]
```

Deletes the cell stack area definition.

Parameters

`-cell` *cell_name*	Specifies the cell(s) for which the cell stack area definition needs to be deleted. *Data_type*: string, optional
`-group` *group_name*	Specifies the group(s) for which the cell stack area definition needs to be deleted. *Data_type*: string, optional
`-help`	Outputs a brief description that includes type and default information for each `delete_cell_stack_area` parameter. For a detailed description of the command and all of its parameters, use the man command: `man delete_cell_stack_area`

Related Information

- `place Category Attributes`
- `report_cell_stack_area`
- `set_cell_stack_area`

delete_cell_stack_group

```
delete_cell_stack_group
[-help]
[-name group_name]
```

Deletes cell stack group definition.

Parameters

`-name` *group_name*	Specifies the group(s) for which the cell stack group definition is to be deleted. *Data_type*: string, optional
`-help`	Outputs a brief description that includes type and default information for each `delete_cell_stack_group` parameter. For a detailed description of the command and all of its parameters, use the man command: `man delete_cell_stack_group`

Related Information

- `place Category Attributes`
- `report_cell_stack_group`
- `set_cell_stack_group`

delete_cell_virtual_align

```
delete_cell_virtual_align
[-help]
[-cells cells]
```

Deletes the cells that need to be virtually aligned.

Parameters

`-cells` *cells*	Specifies the lib cells that require virtual alignment. *Data_type*: (base_cell)+, optional
-help	Outputs a brief description that includes type and default information for each `delete_cell_virtual_align` parameter. For a detailed description of the command and all of its parameters, use the man command: `man delete_cell_virtual_align`

Related Information

- `place Category Attributes`
- `report_cell_virtual_align`
- `specify_cell_virtual_align`

delete_filler

```
   delete_filler
[-cells baseCellNameList]
[-area {x1 y1 x2 y2}]
[-inst instanceName ]
[-prefix prefix ]
[-keep_fixed]
```

Removes physical cell instances, such as filler cells, end-cap cells, and well-tap cells, from standard cell rows.

> Some `delete_filler` parameters can also be specified with `add_fillers` category attributes. However, the parameters you specify with `delete_filler` apply to the current `delete_filler` invocation only, and supersede attributes set by `add_fillers` category. For more information see add_fillers Category Attributes.

Parameters

`-area {x1 y1 x2 y2}`	Specifies the bounding box within which physical instances are deleted. The argument type is a box, in microns. *Data_type*: rect, optional **Note:** Only those filler cells that are completely contained within the area are deleted.
`-cells` *`baseCellNameList`*	Specifies a base cell name or list of base cell names to delete. *Data_type*: string, optional
`-keep_fixed`	Physical instances whose place_status is **FIXED** are not deleted. *Data_type*: bool, optional
`-prefix` *`prefix`*	Delete physical instances whose name starts with *`prefix`*. Does not support wildcards. *Data_type*: string, optional
`-inst` *`instanceName`*	Specifies the name(s) of the instance(s) to delete. You can use wildcards (`*?`) with this parameter. *Data_type*: string, optional

Examples

- The following command deletes all physical-only instances from the design:
  ```
  delete_filler -cells [ get_db -u [ get_db insts -if { .is_physical } ]
  .base_cell.name ] -prefix ""
  ```
- The following command deletes only the instances whose base_cell name is `xyz` and whose names have a `wtap_even` prefix.
  ```
  delete_filler -cells xyz -prefix wtap_even
  ```
- The following command deletes all physical instances with the name prefix `FILLER`:
  ```
  delete_filler -prefix FILLER
  ```
- The following commands delete all physical instances whose base_cell names are `FILL2`, `ecp17`, and `wtp5`:

```
delete_filler -cells "FILL2 ecp17 wtp5" -prefix ""
delete_filler -cells [ list FILL2 ecp17 wtp5 ] -prefix ""
delete_filler -cells {FILL2 ecp17 wtp5} -prefix ""
delete_filler -cells FILL2 ecp17 wtp5 -prefix ""
```

- The following command deletes all physical instances with prefix `WELLTAP` in the specified bounding box of {0 0 300 300} except those instances with place_status `FIXED`:

```
delete_filler -prefix WELLTAP -area {0 0 300 300} -keep_fixed
```

Related Information

- add_fillers
- add_fillers Category Attributes
- check_filler

delete_gate_array_filler_group

```
delete_gate_array_filler_group
[-help]
-group groupName
```

Deletes the gate array filler groups created by `create_gate_array_filler_group`.

Parameters

-help	Outputs a brief description that includes type and default information for each `delete_gate_array_filler_group` parameter. For a detailed description of the command and all of its parameters, use the man command: `man delete_gate_array_filler_group`
-group *groupName*	Specifies the name of the group to be deleted. The * wildcard is supported. *Data_type*: string, required

Related Information

- `create_gate_array_filler_group`
- `place Category Attributes`

delete_inst_padding

```
delete_inst_padding
[-suppress] { inst | -all }
```

Deletes the padding for the specified instances, or all instance-based padding, in the design.

Parameters

`-all`	Deletes padding for all instances. *Data_type*: string, required
`-help`	Outputs a brief description that includes type and default information for each `delete_inst_padding` parameter. For a detailed description of the command and all of its parameters, use the man command: `man delete_inst_padding`
inst	Deletes the padding on the specified instance. *Data_type*: bool, optional
`-suppress`	Suppresses verbose messages. *Data_type*: bool, optional

- The following command deletes padding for one instances named `U27316`:

  ```
  delete_inst_padding U27316
  ```

  It outputs following:

  ```
  Delete Padding for Instant [U27316].
  ```

Note: `delete_inst_padding` deletes padding for either all instances or single instance. Wildcard is not supported yet.

Related Information

- `delete_all_cell_padding`
- `report_inst_padding`
- Placing the Design chapter in the *Innovus Stylus Common UI User Guide*.
    - Adding Padding

delete_inst_space_group

```
delete_inst_space_group
[-help]
-group_name groupName
```

Deletes instance space group by a specific group name.

Parameters

`-group_name` *groupName*	Specifies the name of the instance group. *Data_type*: string, required
`-help`	Outputs a brief description that includes type and default information for each `delete_inst_space_group` parameter. For a detailed description of the command and all of its parameters, use the man command: `man delete_inst_space_group`.

Related Information

- `create_inst_space_group`
- `place Category Attributes`
- `report_inst_space_group`

delete_mim_cap

```
delete_mim_cap
[-help]
[{[-cell cellName] [-prefix prefixName]} [-area x1 y1 x2 y2]]
```

Deletes the mim caps from the design based on cell type or prefix. This command can be used after `add_mim_cap`.

Parameters

`-area x1 y1 x2 y2`	Deletes only the Mim caps that are in the specified box. The `-cell` or `-prefix` options are required when this option is specified. *Data_type*: rect, optional
`-cell cellType`	Deletes the mim caps by cell type. *Data_type*: string, optional
`-help`	Outputs a brief description that includes type and default information for each `delete_mim_cap` parameter. For a detailed description of the command and all of its parameters, use the man command: `man delete_mim_cap`.
`-prefix prefixName`	Deletes the mim caps by prefix. *Data_type*: string, optional

Examples

- The following command deletes the mim cap cell `mim_cell1`:

  ```
  delete_mim_cap -cell mim_cell1
  ```

Related Information

- `add_mim_cap`

delete_pg_keep_out

```
delete_pg_keep_out
[-help]
{-cell lib_cell_name | -inst instance_name | -all }
[-layer layer_name | -layer_id  layer_id]
```

Deletes PG keep out boxes in design for user defined objects.

Parameters

`-all`	Removes all PG keepout boxes in design. *Data_type*: bool, optional
`-cell` *lib_cell_name*	Specifies the lib cell to remove the PG keepout box applied to the specified cell. *Data_type*: string, optional
`-help`	Outputs a brief description that includes type and default information for each delete_pg_keep_out parameter. For a detailed description of the command and all of its parameters, use the man command: `man delete_pg_keep_out`
`-inst` *instance_name*	Specifies the instance to remove the PG keepout box applied to the specified instance. *Data_type*: string, optional
`-layer` *layer_name*	Specifies the names of the layers to delete the PG keepout box applied on the specified metal layers. *Data_type*: string, optional
`-layer_id` *layer_id*	Specifies the layer IDs to delete the PG keepout box applied on the specified metal layer. *Data_type*: string, optional

Related Information

- create_pg_keep_out
- report_pg_keep_out

delete_pipeline_net_group

```
delete_pipeline_net_group
[-help]
-net_group_name netGroupName
```

Deletes pipeline net groups including the subgroups created inside the net groups.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `delete_pipeline_net_group` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man delete_pipeline_net_group`.
`-net_group_name` *netGroupName*	Specifies the name of the net group. *Data_type*: string, optional

Example

- The following command deletes a net group named `netGroup1':
  ```
  delete_pipeline_net_group netGroup1
  ```

Related Information

- `create_pipeline_net_group`
- `place_pipelines`
- `report_pipelines`
- `update_pipeline_net_group`

delete_scan_cell

```
delete_scan_cell
[-help]
cell_name
```

Removes the scan cell specification from a library cell. This command can be called after importing design if there are scan cell specifications in library; or else it can be used after `set_scan_cell`.

Parameters

cell_name	Specifies the cell from which to remove the scan specification. *Data_type*: string, required
-help	Outputs a brief description that includes type and default information for each `delete_scan_cell` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man delete_scan_cell`.

Example

The following command removes the scan specification from the cell named `FF_SCAN`:

```
delete_scan_cell FF_SCAN
```

Related Information

- `delete_all_scan_cells`
- `set_scan_cell`
- `report_scan_cells`
- Placing the Design chapter in the *Innovus Stylus Common UI User Guide*.
    - Optimizing and Reordering Scan Chains

delete_scan_chain

```
delete_scan_chain
[-help]
[-name chainName]
```

Removes a scan chain specification from the design. Use this command if there is scan chain specified or traced.

When you are debugging a new scan chain specification, it is useful to delete all scan chains and start over.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `delete_scan_chain` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man delete_scan_chain`.
`-name` *chainName*	Specifies the name of the scan chain to remove from the design. If you do not specify a scan chain name, the command deletes all scan chains. *Data_type*: string, optional

Example

The following commands load the scan information from a DEF file, remove the scan chain `ABC`, and output the scan information to a DEF file:

```
read_def
delete_scan_chain ABC
write_def -scan_chain
```

Related Information

- `create_scan_chain`
- `create_scan_chain_partition`
- `read_def`
- `report_scan_chain`
- `write_def`
- Placing the Design chapter in the *Innovus Stylus Common UI User Guide.*
    - Optimizing and Reordering Scan Chains

delete_scan_chain_partition

```
delete_scan_chain_partition
[-help]
{-all | -partition partitionName1 partitionName2 ...}
```

Deletes scan chain partitions. The partitions are used to group scan chains so elements in compatible scan chains can be swapped. The scan chains are then reordered on a per-chain basis.

Specify scan chain partitions with the following command: `create_scan_chain_partition`.

Parameters

`-all`	Deletes all partitions from compatible groups. *Data_type*: bool, optional
`-help`	Outputs a brief description that includes type and default information for each `delete_scan_chain_partition` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man delete_scan_chain_partition`.
`-partition` *partitionName1 partitionName2 …*	Specifies the partitions to delete. *Data_type*: string, optional

Example

The following command deletes partition `clk1Domain`:

```
delete_scan_chain_partition clk1Domain
```

Related Information

- `create_scan_chain`
- `create_scan_chain_partition`
- `reorder_scan`
- `reorder_scan Category Attributes`
- `report_scan_chain`
- `report_scan_chain_partitions`
- Placing the Design chapter in the *Innovus Stylus Common UI User Guide*
    - Optimizing and Reordering Scan Chains

delete_spare_module

```
delete_spare_module
[-help]
hinst_name
```

Deletes the instantiation of a created spare module and all the spare instances under the hinst.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `delete_spare_module` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man delete_spare_module`.
`hinst_name`	Specifies the instantiation to delete. *Data_type*: string, required

Example

The following code shows the creation of the spare module SPGM, placement of the module and its instances in the design, and deletion of the hInst of "SPGM_spr_1":

```
#create a spare module
create_spare_module -moduleName SPGM -cell {DFFNSRX2 INVX4 \ NOR2X2} -clock sysclk

#instantiation the spare module and place them in the design
place_spare_modules -moduleName SPGM -channel -minWidth 4.7 -maxWidth 10 \ -minLength 11 -
stepx 10 -stepy 10 -util 0.6

#delete one instantiation of the spare module
delete_spare_module top/SPGM_spr_1
```

Related Information

- `create_spare_module`
- `gui_clear_spare_cells`
- `gui_show_spare_cells`
- `place Category Attributes`
- `place_spare_modules`
- Placing the Design chapter in the *Innovus Stylus Common UI User Guide*.
    - Placing Spare Cells and Spare Modules

gui_clear_scan_chains

```
gui_clear_scan_chains
[-help]
[scanPartitionName]
```

Removes scan chains from the display.

Parameters

scanPartitionName	Specifies the scan partition name to remove from the display. *Data_type*:

Related Information

- Placing the Design chapter in the *Innovus Stylus Common UI User Guide*
  - Optimizing and Reordering Scan Chains

gui_clear_spare_cells

`gui_clear_spare_cells`

Removes spare cell instances from the display.

Parameters

None

Related Information

- `add_spare_insts`
- `gui_show_spare_cells`
- Place Menu chapter of *Innovus Stylus Common UI Menu Reference*.

gui_show_scan_chains

```
gui_show_scan_chains
[-help]
[scan_chain_name]
```

Displays a scan chain. You must load the scan information, place the design, and view the design in the Placement view. Use this command after scan chains are traced.

Parameters

scan_chain_name	Specifies the scan chain. *Default:* If you do not specify this parameter, the software displays all scan chains. *Data_type*: string, optional

Examples

- The following command displays all scan chains in the design:
  ```
  gui_show_scan_chains
  ```
- The following command displays the scan chain whose name is `chain0`.
  ```
  gui_show_scan_chains chain0
  ```

Related Information

- `create_scan_chain`
- `gui_clear_scan_chains`
- `report_scan_chain`
- Placing the Design chapter in the *Innovus Stylus Common UI User Guide*
  - Optimizing and Reordering Scan Chains
- Place Menu chapter of *Innovus Stylus Common UI Menu Reference*.

gui_show_spare_cells

`gui_show_spare_cells`

Displays the results of placing the spare instances. You must specify the spare cell types or spare instances before placing the design.

Parameters

None

Examples

The following set of commands displays all spare instances defined by the `set_spare_insts` command:

```
set_spare_insts -cell ADDFXLT1T10
place_design
gui_show_spare_cells
```

Related Information

- `gui_clear_spare_cells`
- `place_design`
- `set_spare_insts`
- Place Menu chapter of *Innovus Stylus Common UI Menu Reference*.

place_clones

```
   place_clones
[-help]
[-clones cloneinstance(s)orientation(s)]
[-master_partition instance name]
```

Places all clone instances with relative location and relative orientation according to the master partition instances. Adjusts instance orientation based on the clone orientation.

Use this command after running `place_design`.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `place_clones`. For a detailed description of the command and all of its parameters, use the man command: `man place_clones`
`-clones` *`cloneinstance(s)orientation`*	Specifies clone instances with relative location. *Data_type*: list, optional
`-master_partition` *`instance name`*	Specifies the master partition instance. *Data_type*: string, optional

Related Information

- `place_design`
- Place Menu chapter of *Innovus Stylus Common UI Menu Reference*.

place_connected

```
place_connected
[-help]
-attractor macro/IO list
[-fixed]
[-move_fixed]
[-place_status]
{-level integer  | -sequential all_connected | direct_connected | -
stop_points instance_name |-before_points instance_name | -instance_pin instance/pin}
[{-stop_points instance_name | -key_cell cell_name}]
[{-before_points instance_name | -key_cell cell_name}]
[{-instance_pin instance/pin | -key_cell cell_name}]
[ {-attractor_pin pin_list } [-instance_pin instance/pin] ]
```

Places the specified standard cells close to the specified attractor with legal location. The attractor can be IOs, hard macros, fixed standard cell. The recommended flow is to run this command before `place_design`.

Parameters

`-help`	Outputs a brief description that includes type and default information for each place_connected parameter.
`-attractor macro/IO list`	Specifies target hard macros, IOs, or fixed standard cell for attracting standard cells. Wildcards is supported. Tool processes attractors in list one by one. If there are instances shared by different attractors, the locations of shared instances are determined by the later one. In that case, the placement result may vary with different order of objects in list. *Data_type*: string, required
`-attractor_pin pin_list`	Places only standard cells connected with specified pins of attractors close to attractor. Wildcards is supported. Without this setting, standard cells connected with all pins of attractors are pulled. *Data_type*: string, optional

`-before_points` *instance_name*	Specifies that the attracting is stopped before the specified instances. *Data_type*: string, optional
`-fixed`	Sets the placement status as fixed. *Data_type*: bool, optional
`-instance_pin` *instance/pin*	Pulls all instances specified in the list close to the attractor. This option requires only full instance pin name. Wildcards is supported. Option `-attractor_pin` needs to be used with this option. *Data_type*: string, optional
`-key_cell` *cell_name*	Places specified type of cells close to attractors, as well as other logics between the key cell and attractor. Wildcards is supported. Without this setting, all standard cells connected with attractors are pulled. *Data_type*: string, optional
`-level` *integer*	Pulls all instances within certain logic level of connections to the attractor close to attractors. This option is mutually exclusive with `-sequential`, and user must specify one of them. *Data_type*: int, optional
`-move_fixed`	Specifies that instances with the fixed status can also be moved close to the attractor. *Data_type*: bool, optional
`-place_status`	Specifies instances being placed is in “placed” status. By default, the pulled cells are marked as “fixed”. *Data_type*: bool, optional

`-sequential` *`all_connected	direct_connected`*	Pulls sequential cells connected with the attractor close to the attractor. • `all_connected`: attracts all sequential cells from/to attractor; combinational logics between the sequential instance and attractor are attracted as well. • `direct_connected`: attracts sequential cells directly connected with attractor. *Data_type*: enum, optional
`-stop_points` *`instance_name`*	Pulls all instances connected between the attractor and the specified stop point close to attractors. This option requires full instance name or instance pin name. Wildcards is supported. *Data_type*: string, optional

Examples

- The following command places all instances directly connected with all pins of `RAM1` close to `RAM1`.

  ```
  place_connected -attractor RAM1 -level 1
  ```

- The following command pulls all sequential cells from `RAM1` and all combinational cells between `RAM1` and the sequential cells close to `RAM1`.

  ```
  place_connected -attractor RAM1 -sequential all_connected
  ```

- Only the sequential cells from “IN*” pins of `RAM1` and their combinational logics are caught to be placed nearby `RAM1`.

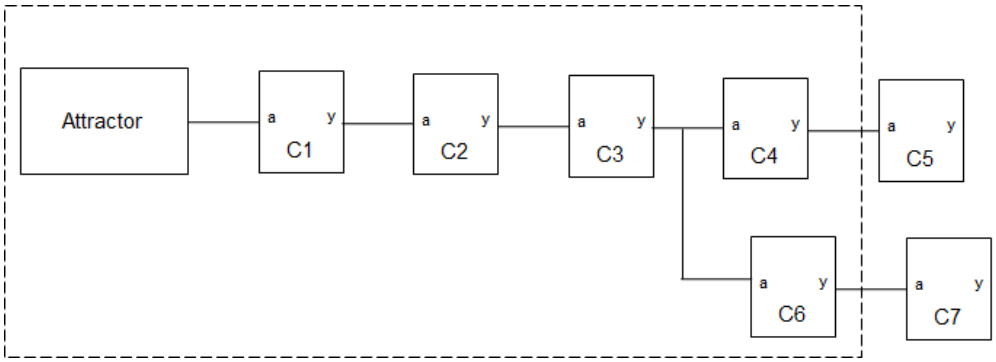
  ```
  place_connected -attractor RAM1 -sequential all_connected -attractor_pin IN*
  ```

- The following command picks out instances that are `"BUF1X"` cell type and directly connected with `RAM1`, and places them close to `RAM1` with `"placed"` status.

  ```
  place_connected -attractor RAM1 -level1 -key_cell BUF1X -place_status
  ```

- The following command places instances, C1, C2, C3, C4, C6, close to the attractor RAM1.

  ```
  place_connected -attractor RAM1 -before_points {C5 C7}
  ```



Related Information

- `place_design`

place_design

```
place_design
[-help]
[-concurrent_macros]
[-incremental]
[-no_pre_place_opt]
[-sdp]
```

Places standard cells based on the global settings for placement, RC extraction, timing analysis, and early global routing. It also relieves the congestion and reorders the scan cells. Also performs the following operations:

- Pre-placement optimization
- Scan tracing and scan reordering
- Placement optimization during the placement process
- Incremental placement on an already placed design

Use the `place_design` command after performing general floorplanning, including macro placement, regioning (if applicable), and power planning on the design. The design could be in the unplaced state, or in a state that the existing placement is to be replaced.

- To run timing-driven placement (the default), make sure the timing constraints are loaded in the design.
- To run scan reorder (the default), specify the scan chain or import the scan DEF data before running this command.

By default, `place_design` performs timing-driven placement with scan reordering, except during prototyping, when it runs in non-timing-driven mode. To change this behavior, run set_db with the following `place` Category attributes prior to running `place_design`:

- `place_global_reorder_scan false`

Or, in prototyping mode:

- `place_design_floorplan_mode`

If there are no timing constraints, `place_design` assumes non-timing-driven mode.

Notes:

- `place_design` supports multiple-CPU processing. For information, see "Multiple-CPU Processing Commands" and "Accelerating the Design Process by Using Multiple-CPU

Processing".

- `place_design` multithreading.
- The `get_metric` command can be used with the `place_design` command to retrieve metrics.
- `place_design` honors the `proto_place_effort` attribute.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `place_design` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man place_design`.
`-concurrent_macros`	When this option is enabled, place_design will do concurrent timing driven placement of macros and standard cells. Also, `place_design -concurrent_macros` by default calls `place_macro_detail` to legalize macros. This parameter works only when the design is already placed. When `place_design -concurrent_macros` is run, it sets the value of the `place_opt_run_global_place` attribute to `seed`. *Data_type*: bool, optional ⚠ When the `set_db place_global_place_io_pins` attribute is set to true, the `place_design -concurrent_macros` command places the placed and unplaced I/O pins in the mixed placer flow and optimize their locations to reduce wire length and congestion. This support is a limited-access feature in this release. It is enabled by a variable specified using the `set_limited_access_feature` command. To use this feature, contact your Cadence representative to explain your usage requirements, and make sure this feature meets your needs before deploying it widely.

`-incremental`	Runs placement incrementally. This parameter works only when the design is already placed. *Data_type*: bool, optional **Note:** Using this parameter defaults to `-no_pre_place_opt`.
`-no_pre_place_opt`	Disables pre-place optimization during the placement run. By default, the command removes all cells with the specified buffer footprint (same as with the `delete_buffer_trees` command), except when using the `-incremental` parameter. *Data_type*: bool, optional
`-sdp`	Enables SDP-aware (Structured Data Path) placement. *Data_type*: bool, optional

Examples

- The following commands disable scan chain reordering:
  ```
  set_db place_global_reorder_scan false
  place_design
  ```
- The following command disables the pre-placed buffer tree removal:
  ```
  place_design -no_pre_place_opt
  ```
- After the design is placed, the following command performs incremental placement, and disables the pre-placed buffer tree removal:
  ```
  place_design -incremental
  ```

Related Information

- `get_metric`
- `place Category Attributes`
- `place_macro_detail`
- `set_db`
- Placing the Design chapter in the *Innovus Stylus Common UI User Guide*

place_detail

```
place_detail
[-area {x1 y1 x2 y2}]
[-eco {true|false}]
[-inst list_of_instances]
```

Corrects flawed cell locations and reports corrections or instance overlap problems. You do not need a fully placed design to run `place_detail`. The `place_detail` command only makes changes if the placement is not legal, such as for instance overlap, incorrect orientation, constraint violations, and placement/routing blockage violations. You can use the `check_place` command to highlight placement violations in the design. The `place_detail` command is called automatically at the end of each major super command such as `place_opt_design`, `opt_design`, and `ccopt_design` to ensure that the final placement is violation-free. If any instances are moved manually, `place_detail` can be run to legalize any cell placements that may have resulted in violations due to this manual movement.

To place unplaced instances, use the `place_eco` command. Use this command after finding an error with the `check_place` command.

`place_detail` only operates on the cells which have the 'placed' placement status.

Note: You can also use `place` attributes to specify parameters for this command. If you use `place` attributes, the parameters apply every time you run `place_detail` in the current session, unless they are overwritten by the `place_detail` settings.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `place_detail` parameter. For a detailed description of the command and all of its parameters, use the man command: `man place_detail`
`-area {x1 y1 x2 y2}`	Specifies the *x1*, *y1, x2*, and *y2* coordinates of the `place_detail` area. *Data_type*: rect, optional **Note**: The coordinates for the `-area` option should be provided in brackets, else the tool will report an error.
`-eco {true|false}`	Activates the ECO mode for the post-route design. *Data_type*: bool, optional
`-inst list_of_instances`	Specifies a list of instances for place_detail to legalize only. *Data_type*: string, optional

Example

- The following command refines the placement in the specified area:

```
place_detail -area {7.925 29.762 11.702 27.531}
*** Starting place_detail (0:04:48 mem=991.9M) ***
Move report: Detail placement moves 0 insts, mean move: 0.00 um, max move: 0.00 um
        Runtime: CPU: 0:00:00.0 REAL: 0:00:00.0 MEM: 991.9MB
Summary Report:
Instances move: 0 (out of 6 movable)
Mean displacement: 0.00 um
Max displacement: 0.00 um
Runtime: CPU: 0:00:00.0 REAL: 0:00:00.0 MEM: 991.9MB
*** Finished place_detail (0:04:48 mem=991.9M) ***
```

Related Information

- `check_place`
- `place_eco`
- Placing the Design chapter in the *Innovus Stylus Common UI User Guide.*
- Place Menu chapter of *Innovus Stylus Common UI Menu Reference*.

place_fix_congestion

```
place_fix_congestion
-help
-area x1 y1 x2 y2
```

Relieves the congestion of the design. This command performs incremental placement based on `route_early_global` congestion results and improves the congestion hot spots by spreading out the cells. It is mostly used after `place_opt_design`, but sometimes it can be used after pre-CTS optimization as well. Since this command tends to move many cells around, it should not be run after post-CTS optimization.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `place_fix_congestion` . For a detailed description of the command and all of its parameters, use the man command: `man place_fix_congestion`
`-area x1 y1 x2 y2`	Specifies the x1, y1, x2, and y2 coordinates of the area in which `place_fix_congestion` will run. **Note**: If you do not specify this option, `place_fix_congestion` will run on the complete design. *Data_type*: rect, optional

Related Information

- `place_opt_design`
- `route_early_global`

place_inst

```
place_inst
[-help]
inst
location
orient
[{-fixed | -placed | -soft_fixed}]
```

Places a leaf instance in the core box. Instance is snapped to the site whose lower-left corner is the nearest to the location specified. If the location specified is outside of the core box, then the instance is snapped to site whose lower-left corner is nearest to the location specified. Instance is not checked for legality–it can overlap other boxes, either placed or fixed, and it can be placed within placement blockages. If no orientation is specified, or if an illegal orientation is specified, `place_inst` will orient the instance with the default orientation of the site in which the instance is being placed. You can check the legality of the placed instance by the `check_place` command.

Note: If you place a master instance blackbox to non-r0 orient, the new non-r0 orient will be converted to r0 orient. For more information, see "Handling of Blackboxes with Non-r0 Orientation" in the Partitioning the Design chapter of the *Innovus Stylus Common UI User Guide*.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `place_inst` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man place_inst`.
`inst`	Specifies the leaf instance name. *Data_type*: (inst), required
`location`	Specifies x and y coordinates of new origin. • *`x`*: Specifies the lower-left x coordinate of the instance on the x axis. • *`y`*: Specifies the lower-left y coordinate of the instance on the y axis. *Data_type*: point, required
`orient`	Specifies the orient of the instance. Specify one of the following values: `r0`, `r90`, `r180`, `r270`, `mx`, `mx90`, `my`, or `my90`. *Default*: `r0` *Data_type*: enum, optional For more information on orient keys, see "Orientation Key" in the Floorplanning the Design chapter of the *Innovus Stylus Common UI User Guide*.
`-fixed | -placed | -soft_fixed`	Specifies that the cell being placed is in the `fixed`, `placed`, or `soft_fixed` status. *Default*: `-fixed` *Data_type*: bool, optional

Example

- The following command places the instance `SH22/I40` with mirrored through Y axis orient at location `1500.3, 980.3` in the floorplan. The instance SH22/I40 is fixed by default and the placement program will not move this instance later:
  ```
  place_inst SH22/I40 1500.3 980.3 my
  ```

- The following command places the instance SH22/I51 with default `R0` orient at location 398.7, 1480.2 in the floorplan. The instance SH22/I52 has 'placed' status and it can be moved during optimization, clock synthesis, or place_detail:

  ```
  place_inst SH22/I51 398.7 1480.2 -placed
  ```

Related Information

- `check_place`
- Floorplanning the Design chapter of the *Innovus Stylus Common UI User Guide*.
- Partitioning the Design chapter of the *Innovus Stylus Common UI User Guide*.

place_jtag

```
place_jtag
[-help]
-row_number nrRow
[-row_top_number nrRowTop]
[-row_bottom_number nrRowBottom]
[-row_left_number nrRowLeft]
[-row_right_number nrRowRight]
[-hard_macro {true | false}]
[-area_io {true | false}]
[-io_net_weight netWtValue]
[-block_net_weight netWtValue]
[-contour {true | false}]
[-ignore_scan {true | false}]
[-orient_top orientation]
[-orient_bottom orientation]
[-orient_left orientation]
[-orient_right orientation]
```

Places JTAG cells. Use this command for JTAG and other cells where placement is required to be close to the perimeter I/Os.

Note: The command does not honor MSMV design.

Parameters

`-area_io {true | false}`	Considers the area around I/O cells as a legal placement location. When specifying this parameter, the `-io_net_weight` parameter is applied to the nets that are connected to the Area I/O cells. *Default*: `false` *Data_type*: bool, optional
`-block_net_weight netWtValue`	Specifies the weight for nets connecting to blocks. This allows `place_jtag` to place more critical block nets in a smaller area. *Default*: `20` *Maximum value:* `512` *Data_type*: int, optional

`-contour {true | false}`	Creates an outline along the boundary of the chip, with its width based on the number of standard cell rows, as specified by the `-row_number` parameter(s). You can specify different values for `-row_top_number`, `-row_bottom_number`, `-row_left_number`, and `-row_right_number`. *Default*: `false` *Data_type*: bool, optional
`-help`	Outputs a brief description that includes type and default information for each `place_jtag` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man place_jtag`.
`-hard_macro {true | false}`	Includes the boundaries of the hard macros in the JTAG placement. *Default*: `false` *Data_type*: bool, optional
`-ignore_scan {true | false}`	Ignores all scan-in pins, scan enable pins, clock pins, and asynchronous control pins. *Default*: `false` *Data_type*: bool, optional ⚠ `place_jtag` relies on the timing library to provide information, such as what pins are scan-in, scan enable, clk, and async control. If the timing library does not have the scan-in and scan enable information, `place_jtag` is not able to ignore them. In this case, before you run `place_jtag`, you must specify those terms using the `set_scan_cell` command.

`-io_net_weight` *netWtValue*	Specifies the weight for nets connecting to I/Os. This allows `place_jtag` to place more critical I/O nets in a smaller area. *Default*: `100` *Maximum*: `512` *Data_type*: int, optional `-io_net_weight` applies to the nets that are connected to any I/O cell, whether it is a standard peripheral I/O cell or an Area I/O cell. This behavior is different from `-block_net_weight`, which specifies the weight of the nets connected to blocks separately from the I/O nets.
`-row_number nrRow`	Specifies the number of standard cell rows to place the JTAG instances. *Data_type*: init, optional *Minimum*: 0 *Maximum*: 10000
`-row_bottom_number` *nrRowBottom*	Specifies the number of standard cell rows at the bottom of the core area to place JTAG logic. *Data_type*: init, optional *Minimum*: 0 *Maximum*: 10000
`-row_left_number` *nrRowLeft*	Specifies the number of standard cell row heights at the left of the core area to place JTAG logic. *Data_type*: init, optional *Minimum*: 0 *Maximum*: 10000
`-row_right_number` *nrRowRight*	Specifies the number of standard cell row heights at the right of the core area to place JTAG logic. *Data_type*: init, optional *Minimum*: 0 *Maximum*: 10000

`-row_top_number` *nrRowTop*	Specifies the number of standard cell rows at the top of the core area to place JTAG logic. *Data_type*: init, optional *Minimum*: 0 *Maximum*: 10000
`-orient_bottom` *orientation*	Considers the specified orientation at each side first and then looks for other legal orientations. Specify one of the following values: `R0 R90 R180 R270 MX MX90 MY MY90`. *Data_type*: string, optional
`-orient_left` *orientation*	Considers the specified orientation at each side first and then looks for other legal orientations. Specify one of the following values: `R0 R90 R180 R270 MX MX90 MY MY90`. *Data_type*: string, optional
`-orient_right` *orientation*	Considers the specified orientation at each side first and then looks for other legal orientations. Specify one of the following values: `R0 R90 R180 R270 MX MX90 MY MY90`. *Data_type*: string, optional
`-orient_top` *orientation*	Considers the specified orientation at each side first and then looks for other legal orientations. Specify one of the following values: `R0 R90 R180 R270 MX MX90 MY MY90`. *Data_type*: string, optional

Examples

The following command places the JTAG instances within four standard cell rows at the top and bottom of the chip and within four times the standard cell row height at the sides of the chip:

```
place_jtag -row_number 4
```

The following command places the JTAG instances within ten standard cell rows at the top and bottom of the chip and within ten times the standard cell row height at the sides of the chip. It increases the I/O net weight to `256`.

```
place_jtag -row_number 10 -io_net_weight 256
```

Related Information

- `reset_jtag`
- `set_jtag`
- `set_scan_cell`
- `set_jtag_to_unplaced`
- `trace_jtag`
- Placing the Design chapter in the *Innovus Stylus Common UI User Guide*
  - Specifying and Placing JTAG and Other Cells Close to the I/Os
- Place Menu chapter of *Innovus Stylus Common UI Menu Reference*.

place_opt_design

```
place_opt_design
[-help]
[-expanded_views]
[-incremental]
[-incremental_timing]
[-num_paths number_of_paths]
[-report_dir outputDirectory]
[-report_prefix outputFileName]
[-timing_debug_report]
```

Executes pre-CTS flow with both placement and pre-CTS optimization.

Earlier, users had to perform the following two steps to do placement and pre-CTS optimization:

1. `place_design`
2. `opt_design -pre_cts`

With `place_opt_design` command, users need to run only a single command to do placement and pre-CTS optimization.

Combining the two commands, `place_design` and `opt_design -pre_cts`, provides better integration between placement and optimization to achieve faster runtime and better PPA.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `place_opt_design` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man place_opt_design`.
`-expanded_views`	Prints timing information for each active view at the end of the command restoring all pruned views if absent. *Data_type*: bool, optional

`-incremental`	Specifies `place_opt_design` to be run in the incremental mode. In the incremental mode `place_opt_design` incrementally improves the existing placement based on the critical timing paths and congestion. The command then re-optimizes the netlist based on the new placement. The optimization in incremental `place_opt_design` does DRV fixing, WNS, and TNS optimization. The input to `place_opt_design -incremental` can be either a partial placement or full placement with placement violations, and the output is a legal placement as `refinePlace` is called at the end. This can be used if you want to further improve the congestion and timing QoR or if there is a minor floorplan change and this by default, congestion-driven, and timing-driven. It also honors density screens and floorplan constraints. If you run `place_opt_design` and notice some areas of congestion, you can introduce density screens and run `place_opt_design -incremental`. As `place_opt_design -incremental` honors density screens, it would spread out some of the cells, reducing congestion in that area. In contrast, `refinePlace` would not do this as it only works on the hard placement violations such as cell overlaps, while density screens is a "soft" constraint. There is no scan-reordering done in `place_opt_design -incremental`. **Note**: Early clock flow mode `place_opt_design -incremental` reuses the existing early clock flow clock tree without any logical or physical modification and makes adjustments only to useful skew using virtual delays. It is also essential to note that the CCOpt clock tree spec must not be deleted between the original and incremental `place_opt_design` steps, just like it must not be done between `place_opt_design` and `ccopt_design`, otherwise any useful skew will be lost. *Data_type*: bool, optional

`-incremental_timing`	In the `incremental_timing` mode, `place_opt_design` does incremental DRV and timing optimization. In this mode, `place_opt_design` incrementally optimizes the critical timing path. Also, the optimization in incremental timing `place_opt_design` does DRV fixing, WNS, and TNS optimization. *Data_type*: bool, optional
`-num_paths` *number_of_paths*	Specifies the number of paths reported in the timing report. By default, all timing paths are reported. *Data_type*: integer, optional
`-report_dir` *outputDirectory*	Specifies the output directory in which the timing/drv reports will be saved. *Data_type*: string, optional
`-report_prefix` *outputFileName*	Specifies the file name prefix for the timing/drv reports that are generated at the end of `place_opt_design`. *Data_type*: string, optional
`-timing_debug_report`	When specified, generates a machine-readable timing debug report after completing placement and preCTS optimization. This report is used for debugging timing results using the timing debug feature in Stylus. **Note**: You can specify the number of paths to be reported using the `-num_paths` parameter. *Data_type*: bool, optional

Related Information

- `opt_design`
- `place_design`
- `place_fix_congestion`
- Placing the Design chapter in the *Innovus Stylus Common UI User Guide*

place_pad

```
place_pad
[-help]
pad
[orient]
{x y}
```

Places an I/O pad instance. This command can be used before the `place_design` command.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `place_pad` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man place_pad`.
`{x y}`	Coordinates of new origin. The option is a point and a must. *Data_type*: point, required
`orient`	Specifies the orient of the pad cell. Specify one of the following values: `r0`, `r90`, `r180`, `r270`, `mx`, `mx90`, `my`, and `my90`. *Default:* `r0` *Data_type*: string, optional
`pad`	Specifies the pad instance name. *Data_type*: string, required

Example

The following command places the I/O pad instance with mirrored axis orient, rotated 90 degrees counter clockwise:

```
place_pad SH14/I412 498.3 5000.8 mx90
```

Related Information

- place_design

place_pipelines

```
place_pipelines
[-net_groups {netGroupName}]
[-region {true | false }]
[-status {fixed | placed}
[-timing]
[-density utilization_value]
```

Places the registers of the specified pipeline flip-flops (PFF) net group(s) so that the distances between the stages of each group are roughly equal. This command can be used if the pipeline net groups are specified in the design.

Parameters

`-net_groups {`*`netGroupName`*`}`	Specifies the PFF net group, or a list of net groups, to place. By default, all PFF net groups in design are considered. *Data_type*: string, optional
`-region {true | false}`	Creates region constraints for flops in each stage of pipeline, to allow adjustments at a later stage. *Default:* `false` *Data_type*: bool, optional
`-status {fixed | placed}`	Places the PFFs with fixed placement status, or placed status. *Default:* `placed` *Data_type*: enum, optional
`-timing`	Automatically finds reserve lengths. The option is of type boolean and is optional. *Data_type*: bool, optional
`-density` *`utilization_value`*	Specifies the expected utilization of regions for flops. *Minimum:* `0` *Maximum:* `1` *Data_type*: float, optional

Example

- The following command places the registers of all pipeline flip-flops (PFF) net groups at equal distance:

  ```
  place_pipelines
  ```

- The following command places the registers of pipeline flip-flops (PFF) net group named `netGroup1` at equal distance and marked the registers status `FIXED`:

  ```
  place_pipelines -net_groups netGroup1 -status fixed
  ```

Related Information

- `create_pipeline_bus_guide`
- `create_pipeline_net_group`
- `delete_pipeline_net_group`
- `report_pipelines`
- `update_pipeline_net_group`

place_spare_modules

```
place_spare_modules
-module_name moduleName
{-num_modules integer | -step_x stepDistance -step_y stepDistance}
[-area x1 y1 x2 y2]
[-channel
    {-max_width maxWidth}
    {-min_width minWidth}
    {-min_length minLength}]
[-offset_x offsetDistance]
[-offset_y offsetDistance]
[-power_domain powerDomainName]
[-prefix prefix]
[-density utilizationFactor]
```

Instantiates a created spare module, and places spare modules across the design. You can run this command before or after placement. If you run it after placement, it calls `place_eco`.

Parameters

`-area x1 y1 x2 y2`	Specifies box coordinates (x1, y1, x2, and y2) in micron. *Data_type*: rect, optional
`-channel`	Places spare modules in thin channels between placement blockages. *Data_type*: bool, optional
`-max_width maxWidth`	Specifies the maximum channel width, in microns. The `maxWidth` determines whether the placeable area between two macro blocks is considered a channel. *Data_type*: float, optional
`-min_length minLength`	Specifies the minimum channel length, in microns. The `minLength` must be greater than the specified `maxWidth`. *Data_type*: float*, optional*

`-min_width` *minWidth*	Specifies the minimum channel width, in microns. The *minWidth* should be greater than the width of the largest-space cell provided. *Data_type*: float, optional
`-module_name` *moduleName*	Specifies the name of the spare module. This module must have been created using the `create_spare_module` command. *Data_type*: string, required
`-num_modules` *integer*	Specifies the total number of spare modules to add. The software calculates how far apart to place the modules. *Data_type*: int, optional
`-offset_x` *offsetDistance*	Specifies the offset along the x axis, in microns, from the left core boundary. *Data_type*: float, optional
`-offset_y` *offsetDistance*	Specifies the offset along the y axis, in microns, from the bottom core boundary. *Data_type*: float, optional
`-power_domain` *powerDomainName*	Specifies the power domain in which to place the spare modules. *Data_type*: string, optional
`-prefix` *prefix*	Specifies a prefix to add to the automatically generated prefix for spare modules. The software already uses the prefix spr_*n* for spare modules. If you specify `-prefix spareMod`, the instances of this module are named `spareMod_spr_1, spareMod_spr_2, spareMod_spr_3`, and so on. *Data_type*: string, optional
`-step_x` *stepDistance*	Specifies the distance, in microns, between the instantiated spare modules along the x axis. The *stepDistance* must be greater than the spare module bounding box. *Data_type*: float, optional

`-step_y` *`stepDistance`*	Specifies the distance, in microns, between the instantiated spare modules along the y axis. The *`stepDistance`* must be greater than the spare module bounding box. *Data_type*: float, optional
`-density` *`utilizationFactor`*	Specifies the utilization factor of the spare modules. *Value Range:* Greater than 0.0; less than 1.0. *Data_type*: float, optional

Note: The `place_spare_modules` command does not honor `-density` when `-num_modules` switch is used.

Command Order

Run this command after running `create_spare_module`.

Examples

The following command instantiates module `mod1` and places the instantiations at intervals of 50 microns along the x and y axis. It places the instantiations at an offset of 5 microns from the left core boundary and the bottom core boundary:

```
place_spare_modules -module_name mod1 -step_x 50 -step_y 50 -offset_x 5 -offset_y 5
```

The following commands create spare module SPGM with cells DFFNSRX2, INVX4, and NOR2X2 using clock net sysclk; then instantiates module SPGMand places the instantiations between channels that are between 4.7 um and 10.0 um wide and at least 11.0 um long. The step size is 10 um in the x direction and 10 um in the y direction and the maximum utilization rate is 60 percent:

```
create_spare_module -module_name SPGM -cell {DFFNSRX2 INVX4 \
  NOR2X2} -clock sysclk
place_spare_modules -module_name SPGM -channel -min_width 4.7 -max_width 10 \
  -min_lengthgth 11 -step_x 10 -step_y 10 -density 0.6
```

Related Information

- `create_spare_module`
- `delete_spare_module`
- `place_eco`
- Placing the Design chapter in the *Innovus Stylus Common UI User Guide*
  - Placing Spare Cells and Spare Modules
- Place Menu chapter of *Innovus Stylus Common UI Menu Reference*.

place Category Attributes

The following types of place category attributes are covered in this chapter:

- place_cell_edge_spacing
- place_design_floorplan_mode
- place_design_refine_macro
- place_design_refine_place
- place_detail_activity_power_driven
- place_detail_allow_border_pin_abut
- place_detail_allow_single_height_row_symmetry_x
- place_detail_check_cut_spacing
- place_detail_check_inst_space_group
- place_detail_check_route
- place_detail_eco_max_distance
- place_detail_eco_priority_insts
- place_detail_fixed_shifter
- place_detail_honor_inst_pad
- place_detail_irdrop_aware_effort
- place_detail_io_pin_blockage
- place_detail_ir_aware_max_drive_strength
- place_detail_irdrop_aware_timing_effort
- place_detail_irdrop_region_number
- place_detail_legalization_inst_gap
- place_detail_m3_stripe_push_down
- place_detail_m3_stripe_shrink
- place_detail_max_shifter_depth
- place_detail_max_shifter_column_depth
- place_detail_max_shifter_row_depth

- place_detail_no_filler_without_implant
- place_detail_use_gate_array_filler_groups
- place_detail_use_no_diffusion_one_site_filler
- place_detail_context_aware_legal
- place_detail_color_aware_legal
- place_detail_pad_fixed_insts
- place_detail_pad_physical_cells
- place_detail_preroute_as_obs
- place_detail_preserve_routing
- place_detail_remove_affected_routing
- place_detail_sdp_alignment_in_refine
- place_detail_swap_eeq_cells
- place_detail_use_check_drc
- place_detail_use_diffusion_transition_fill
- place_detail_wire_length_opt_effort
- place_global_activity_power_driven_effort
- place_global_align_macro
- place_global_allow_3d_stack
- place_global_clock_power_driven_effort
- place_global_activity_power_driven
- place_global_auto_blockage_in_channel
- place_global_clock_gate_aware
- place_global_clock_power_driven
- place_global_cong_effort
- place_global_cpg_effort
- place_global_cpg_file
- place_global_enable_advanced_pipeline
- place_global_enable_distributed_place

- place_global_ignore_scan
- place_global_ignore_spare
- place_global_max_density
- place_global_module_aware_spare
- place_global_module_padding
- place_global_place_io_pins
- place_global_sdp_alignment
- place_global_sdp_place
- place_global_reorder_scan
- place_global_soft_guide_strength
- place_global_timing_effort
- place_global_uniform_density
- place_hard_fence
- place_opt_post_place_tcl
- place_opt_run_global_place
- place_spare_update_timing_graph

To set the value of any root attribute, use the `set_db` command:

```
set_db attribute_name value
```

To get the current value of any root attribute, use the `get_db` command:

```
get_db attribute_name
```

To get the current value of all attributes in the `place` category, use the following command:

```
get_db -category place
```

place_cell_edge_spacing

`place_cell_edge_spacing {cell_edge_type1 cell_edge_type2 spacing}`

Default: `""`

Data_type: string, read-only

Specifies the list of cell edge-type spacing values from `LEF CELLEDGESPACINGTABLE` or OA techfile that

affects the placer. It specifies the min-spacing between different edge-types of std-cells, in a list format: `{cell_edge_type1 cell_edge_type2 spacing}`. The edge-type names match the base_cell's right_edge_type and left_edge_type attribute values.

place_design_floorplan_mode

`place_design_floorplan_mode {true | false}`

Default: `false`

Data_type: bool, read/write

Runs placement in the floorplan mode. This mode is used for prototyping and runs quickly to gauge the feasibility of the netlist, but might not place design components in legal locations.

Note: When `place_design_floorplan_mode` is set to `true`, it disables congestion-driven placement. Attribute `place_global_cong_effort` is set to low.

place_design_refine_macro

`place_design_refine_macro  {true | false}`

Default: `false`

Data_type: bool, read/write

Specifies whether to run `place_macro_detail` after `place_design -concurrent_macros`. When `place_design -concurrent_macros` is run, it sets the value of `place_opt_run_global_place` attribute to `seed`.

place_design_refine_place

`place_design_refine_place {true | false}`

Default: `true`

Data_type: bool, read/write

Specifies whether to run detailed placement (`place_detail`) at the end of `place_design`, to legalize placed instances.

place_detail_activity_power_driven

```
place_detail_activity_power_driven {true | false}
```

Default: `false`

Data_type: bool, read/write

When `true`, it turns on the activity-driven tweakage, and flow change depends on the effort.

If the `design_power_effort` attribute is set to `high`, `place_design_refine_place` is automatically set to `true`.

place_detail_allow_border_pin_abut

```
place_detail_allow_border_pin_abut {true | false}
```

Default: `false`

Data_type: bool, read/write

Some libraries have cells with pin geometry that is not set back by half-routing pitch from the normal PR-boundary of that cell. In cases where the pin geometries touch the cell PR_boundary, placement of an instance of this cell can cause a DRV violation with a normal pin geometry of the other cells. To avoid these DRVs, placer will add one Site padding to the cell-side during pre-processing, thus satisfying the min-spacing rules. The `place_detail_allow_border_pin_abut` attribute disables this pre-emptive DRV avoidance behavior, and no auto-padding is done when the option is set to `true`.

place_detail_allow_single_height_row_symmetry_x

```
place_detail_allow_single_height_row_symmetry_x siteName
```

Default: `""`

Data_type: bool, read/write

Specifies a single height row tech-site, which allows symmetry X during detail place.

place_detail_check_cut_spacing

```
place_detail_check_cut_spacing {true | false}
```

Default: `false`

Data_type: bool, read/write

Enables the via-to-via spacing check during `place_detail` and `check_place`. The checking is limited to single-cut via only. Via-array spacing is not supported yet.

This is turned on by default for 16nm and below process designs.

Related Commands

- `check_place`

place_detail_check_inst_space_group

```
place_detail_check_inst_space_group {true | false}
```

Default: `false`

Data_type: bool, read/write
Specifies whether the distance between instances that belong to the same space group would be honored during placement.

Related Commands

- `check_place`

place_detail_check_route

```
place_detail_check_route {true | false}
```

Default: `false`

Data_type: bool, read/write

Considers pre-routed FIXED signal wires and avoids creating DRC violations between FIXED signal wires and instance pins. When this attribute is not specified, place_detail ignores FIXED signal wires.

Note: Using this attribute might result in longer run time.

If you specify `place_detail -io_pin_blockage`, you do not need to specify `-check_route` for signal pins because `place_detail` will treat the signal pins similar to supply pins for blockages. However, signal wires still need this attribute.

place_detail_eco_max_distance

`place_detail_eco_max_distance` *micron*

Default: `0.0`

Data_type: double, read/write

Specifies max distance, in microns, in the horizontal or vertical direction for the `place_detail` ECO mode.

For example, if the original location of the instance is at (x0 y0) and the specified eco max distance is M, the search window will be (x0-M, yo-M) (x0+M y0+M). The maximum movement should be less than 2M.

If `place_detail_eco_max_distance` is not defined by the user, and the user does `place_detail -eco`, the value of `place_detail_eco_max_distance` is internally set to 10 Standard Cell height units.

Note: This option will impact both `place_detail -eco` and `place_detail -inst`.

place_detail_eco_priority_insts

`place_detail_eco_priority_insts {placed|fixed|eco}`

Default: `placed`

Data_type: enum {placed fixed eco}, read/write

Specifies the instance priority for `place_detail` ECO mode.

- `placed`: Treats current instances as placeable but with higher priority.
- `fixed`: Treats placed instances as fixed.
- `eco`: Treats ECO instances with higher priority.

place_detail_fixed_shifter

`place_detail_fixed_shifter {true | false}`

Default: `false`

Data_type: bool, read/write

Specifies whether to set the shifters to `FIXED` status after placing them.

When set to `true`, the level shifters are `FIXED` after placement. However, the placement for some of the

level shifters can be skipped due to the following reasons:

- The shifter has no connections outside its power domains.
- The shifter has no valid location.

place_detail_honor_inst_pad

```
place_detail_honor_inst_pad {true | false}
```

Default: `false`

Data_type: bool, read/write

Honors padding from `set_inst_padding` in detail placement.

place_detail_irdrop_aware_effort

```
place_detail_irdrop_aware_effort {none low medium high}
```

Default: `none`

Data_type: enum {none low medium high}, read/write

Used to enable the IR-aware feature in refinePlace. You can use the following settings to specify IR-drop optimization effort level to be used during `place_detail`.

You can specify the following optimization effort levels:

- `none`: Disables the feature, default value.
- `low`: Moves the IR-violating instances to nearby rows (in the vertical direction). The aggressor has the same priority as the other instances during legalization.
- `medium`: : Moves the IR violating instances in both horizontal and vertical directions. The aggressor has higher priority than the other instances during legalization.
- `high`: Moves the aggressor instances to nearby rows. The aggressor has higher priority than the other instances during legalization and considers the IR-drop violation as a hard rule during legalization. This option is not recommended in this version.

place_detail_io_pin_blockage

```
place_detail_io_pin_blockage {true | false}
```

Default: `false`

Data_type: bool, read/write

Treats ioPins from top-level as pre-routes during DRV checks in legalization.

place_detail_ir_aware_max_drive_strength

`place_detail_ir_aware_max_drive_strength` *`maxDriveStrength`*

Default: `0.0`

Data_type: double, read/write

Specifies the current density value for the IR critical instances above which the instances gets padded during `place_detail`. This means that the IR critical instances with the current density value more than the specified value gets padded.

Note: There may be larger cell movement with this option, which can degrade the timing. So, it is preferred to be used at the pre-CTS stage in the flow. The current density values for the IR-critical instances are dumped in the Voltus Reports.

place_detail_irdrop_aware_timing_effort

`place_detail_irdrop_aware_timing_effort {none|standard|high}`

Default: `high`

Data_type: enum, read/write

Specifies the timing-effort level to be used during IR-aware `place_detail`.

You can specify the following timing effort levels:

- `none`: Disables the feature, `place_detail` is not timing aware. Tool uses the default value.
- `medium`: `place_detail` is timing aware and minimizes the movement of the timing sensitive instances during `place_detail`.
- `high`: `place_detail` is timing aware and does not move the timing sensitive instances during `place_detail`.

place_detail_irdrop_region_number

`place_detail_irdrop_region_number` *`valueOfDebugIrdrop`*

Default: `100`

Data_type: int, read/write

Specifies the total number of regions in which the tool identifies the IR-violating instances. The tool dumps the IR-violating instances inside these regions in the reports.

place_detail_legalization_inst_gap

`place_detail_legalization_inst_gap` *`minimum_gap`*

Default: `0`

Data_type: int, read/write

Specifies the minimum gap between instances (unit sites).

place_detail_m3_stripe_push_down

`place_detail_m3_stripe_shrink` *`site_units`*

Default: `-1`

Data_type: int

Specifies the width threshold in site units for virtual push down of M3 stripe to M2 for M1 pin-access.

If the default value is set to `-1`, it indicates that no M3 stripe will be pushed down. If the default value is set to `0`, it indicates that all M3 stripes will be pushed down. If the default value is larger than `0`, then all M3 stripes having width greater than the specified value will be pushed down.

place_detail_m3_stripe_shrink

`place_detail_m3_stripe_shrink` *`site_units`*

Default: `0`

Data_type: int, read/write

Specifies the shrink effort of the virtual M2 stripe by given total site units. This global should work with place_detail_m3_stripe_push_down.

If the default value is set to `0`, it indicates that the entire pushed down M3 stripe is used to evaluate M1 pin access. If the default value is set to more than `0`, it indicates that M3 stripe will shrink by the specified value before it is pushed down to M2. The left side will shrink first. For example, if the specified value is set to `1`, the M3 stripe will shrink by 1 site on left, and if the specified value is set to `2`,

1 site on left will shrink, along with 1 site on right.

place_detail_max_shifter_depth

`place_detail_max_shifter_depth` *microns*

Default: `999` (no effect)

Data_type: double

Specifies the maximum distance from a horizontal or vertical side that a shifter can be placed.

place_detail_max_shifter_column_depth

`place_detail_max_shifter_column_depth` *microns*

Default: `999.0` (no effect)

Data_type: double, read/write

Specifies the maximum distance from a vertical side that a shifter can be placed.

place_detail_max_shifter_row_depth

`place_detail_max_shifter_row_depth` *microns*

Default: `999` (no effect)

Data_type: double, read/write

Specifies the maximum distance from a horizontal side that a shifter can be placed. The distance is measured from the power domain boundary, not from the edge of the placeable areas and accounts for gaps.

place_detail_no_filler_without_implant

`place_detail_no_filler_without_implant {true | false}`

Default: `false`

Data_type: bool

Specifies that there is no cell without the implant obs. For designs in which all standard cells and fillers have implant obs, you can specify this option to preserve enough spacing during `place_detail` for later filler insertion.

place_detail_use_gate_array_filler_groups

```
place_detail_use_gate_array_filler_groups {true | false}
```

Default: `false`

Data_type: bool, read/write

Specifies that the filler cell instances will only be replaced by logic instances that belong to same GA groups.

place_detail_use_no_diffusion_one_site_filler

```
place_detail_use_no_diffusion_one_site_filler {true | false}
```

Default: `false`

Data_type: bool, read/write

Specifies that all one-site fillers do not have diffusion obs. For designs in which all one-site fillers without diffusion obs, you can use this option to preserve enough spacing during refinePlace for later filler insertion. This option can override the setting of `place_detail_legalization_inst_gap` to `false`.

place_detail_context_aware_legal

p`lace_detail_context_aware_legal {none|all|optional|required|user|ignore_soft}`

Default: `all`

Data_type: enum* {none all optional required user ignore_soft}, read/write

Instructs placer to support cell edge to cell edge spacing constraints as defined in the LEF files.

The `optional`, `required`, `user`, `ignore_soft` argument values can be used separately or together, but they are exclusive to `all` and `none`.

place_detail_color_aware_legal

```
place_detail_color_aware_legal {true | false}
```

Default: `false`

Data_type: bool, read/write

Enables placer to honor and fix double pattern constraints between adjacent cells.

Note: This attribute does not affect cell to pre-route color DRC checks.

place_detail_pad_fixed_insts

`place_detail_pad_fixed_insts {true | false}`

Default: `false`

Data_type: bool, read/write

Specifies whether the `place_detail` command honors cell padding on `FIXED` instances.

place_detail_pad_physical_cells

`place_detail_pad_physical_cells {true | false}`

Default: `false`

Data_type: bool, read/write

Honors padding overlap with physical cells.

place_detail_preroute_as_obs

`place_detail_preroute_as_obs`

Default: `""`

Data_type: string, read/write

Specifies layers on which preroute are taken as OBS.

You can change this behavior by using the following command before running placement:

`set_db place_detail_preroute_as_obs`

Note: This attribute is not recommended for Advanced nodes of 20nm and below.

place_detail_preserve_routing

`place_detail_preserve_routing {true | false}`

Default: `true` (`place_detail` deletes all routed wires.)

Data_type: bool, read/write

Preserves all routed wires. This option does not effect the behavior of `place_design`, but is applicable in calls to `place_detail`, either stand-alone, or internal to `opt_design`.

place_detail_remove_affected_routing

```
place_detail_remove_affected_routing {true | false}
```

Default: `false`

Data_type: bool, read/write

Specifies whether `place_detail` removes wires connected to moved cells only or to all cells.

Removes all wires, whether connected cells were moved or not.

place_detail_sdp_alignment_in_refine

```
place_detail_sdp_alignment_in_refine {true | false}
```

Default: `false`

Data_type: bool, read/write

Enables SDP legalization/alignment for each `place_detail` run during detailed cell placement.

place_detail_swap_eeq_cells

```
place_detail_swap_eeq_cells {true | false}
```

Default: `false`

Data_type: bool, read/write

Determines whether master cells can be replaced by EEQ cells during detailed placement to improve routability. After detailed placement, the software reports the number of replacements. EEQ cells are defined in the LEF file.

place_detail_use_check_drc

```
place_detail_use_check_drc {true | false}
```

Default: `false`

Data_type: bool, read/write

Enables the FGC based DRC engine in legalization.

Enables the FGC engine inside placement for DRC checking. This DRC engine is the same as one used by NanoRoute and `check_drc`.

place_detail_use_diffusion_transition_fill

```
place_detail_use_diffusion_transition_fill {true | false}
```

Default: `false`

Data_type: bool, read/write

Specifies that the design has fillers with jogged diffusion obs. Use this attribute for designs where fillers with jogged diffusion obs are to be inserted between cells with varying diffusion widths.

place_detail_wire_length_opt_effort

```
place_detail_wire_length_opt_effort {none | medium | high}
```

Default: `medium`

Data_type: enum {none medium high}, read/write

Optimizes wire length by swapping cells. Optimization can reduce the total wire length up to two percent without degrading placement. In high-effort mode, it might add up to ten percent to the run time.

place_global_activity_power_driven_effort

```
place_global_activity_power_driven_effort {standard|high}
```

Default: `standard`

Data_type: enum {standard high}, read/write

Specifies `place_global_activity_power_driven_effort` mode.

place_global_align_macro

```
place_global_align_macro {true|false}
```

Default: `true`

Data_type: bool, read/write

Enables mixed-placer to align nearby same-size macros during the global placement.

place_global_allow_3d_stack

`place_global_allow_3d_stack {true|false}`

Default: `false`

Data_type: bool, read/write

Enables the 3D placement.

place_global_clock_power_driven_effort

`place_global_clock_power_driven_effort {none|standard|high}`

Default: `low`

Data_type: enum {low standard high}, read/write

Specifies the `place_global_clock_power_driven_effort` mode.

place_global_activity_power_driven

`place_global_activity_power_driven {true|false}`

Default: `false`

Data_type: bool, read/write

Identifies and constrains power-critical nets to reduce switching power.

place_global_auto_blockage_in_channel

`place_global_auto_blockage_in_channel {none | soft| partial}`

Default: `none`

Data_type: enum {none soft partial}, read/write

Creates placement blockages in narrow channels between macros before Global Placement

- `soft`: Creates a soft blockage automatically by calling the `finish_floorplan` command.
- `partial`: Creates a partial blockage whose density will be determined automatically.

- `none`: Prevents the creation of blockage automatically by placer.

Note: The "soft" option only prevents cells being placed inside the channels during global placement. place_detailr called by optimization can still place the cells inside the channels.

Note: The threshold to activate the soft blockage insertion when the option is set to "soft" is when the channels between macros are less than 50 um in width.

place_global_clock_gate_aware

```
place_global_clock_gate_aware {true | false}
```

Default: `true`

Data_type: bool, read/write

Specifies that placement is aware of clock gate cells in the design.

Note: `-place_global_clock_gate_aware` finds better placement for clock gating elements towards the center of gravity for fanout.

place_global_clock_power_driven

```
place_global_clock_power_driven {true | false}
```

Default: `true`

Data_type: bool, read/write

Improves clock wirelength and subsequently clock power by adding more weight to clock nets between clock gates and their respective Flops.

place_global_cong_effort

```
place_global_cong_effort {low | medium | high | extreme |auto}
```

Default: `auto`

Data_type: enum {low medium high extreme auto}, read/write

Specifies the effort level for relieving congestion.

Specify one of the following values:

- `auto`: Automatically determines whether the design is congested and performs extra congestion driven effort for highly congested designs.

- `high`: Runs more iterations of placement in an effort to achieve better congestion results. This parameter increases the run time.
- `low`: Runs fewer iterations of placement to arrive quickly at a legal placement. This parameter might decrease placement quality.
- `medium`: Runs placement on designs at a normal effort level.

place_global_cpg_effort

```
place_global_cpg_effort {low | medium | high}
```

Default: `low`

Data_type: enum {low medium high}, read/write

Specifies the effort level of pushing instances to the guided locations during the CPG flow.

With higher level, placement will increase the force to push instances to the guided locations, but might degrade wire length and timing QOR.

place_global_cpg_file

```
place_global_cpg_file cpgFileName
```

Default: `""`

Data_type: string, read/write

Specifies the CPG guide file. When this option is specified, the CPG flow is turned ON automatically.

Note: To run the CPG flow, you must save a guide file from the old version database and read it during the mixed placement flow on the new version netlist.

place_global_enable_advanced_pipeline

```
place_global_enable_advanced_pipeline {true | false}
```

Default: `false`

Data_type: bool, read/write

Enables advanced pipeline placement.

place_global_enable_distributed_place

```
place_global_enable_distributed_place {true | false}
```

Default: `false`

Data_type: bool, read/write

Enables distributed placement platform.

Note: This attribute is part of a limited-access feature in this release. It is enabled by a variable specified using the `set_limited_access_feature` command. To use this feature, contact your Cadence representative to explain your usage requirements, and make sure this feature meets your needs before deploying it widely.

place_global_ignore_scan

```
place_global_ignore_scan {true | false}
```

Default: `true`

Data_type: bool, read/write

Global placement disregards scan connections while placing scan groups. Before you place the design, you must already have specified the scan cells with the `set_scan_cell` command, or the scan cell information must be in the timing library.

Note:

- This attribute cannot skip scan definition checking. No matter whether the option is false or true, scan chain definition must be imported before the global placement if the netlist has a scan connection.
- Scan groups are sometimes called scan partitions.

place_global_ignore_spare

```
place_global_ignore_spare {true | false}
```

Default: `false`

Data_type: bool, read/write

Disregards spare cell connections while placing spare cells. Before you place the design, you must already have specified the spare cells with the `set_spare_insts` command, or the spare cell information must be in the timing library.

`place_design` ignores spare cell connections.

Note: The placer disconnects high-fanout (nets with more than 75 terminals) from spare cells, even when this parameter is set to `0`.

place_global_max_density

`place_global_max_density` *value*

Default: `-1.0`

Data_type: double, read/write

Controls maximum density of local bins during global placement. Sets the maximum density of core area so that the placement engine aggressively minimizes the wirelength, while satisfying the maximum density constraint. This helps to achieve better timing results in low utilization designs. After global placement, the density control is removed.

Use a value between 0 (zero percent) and 1 (100 percent). For example, a value of 0.5 indicates that the maximum density in the entire design will be 50 percent or less.

Note: It is recommended that the `place_global_max_density` value should be greater than standard cell utilization of design.

Note: Using this attribute might degrade congestion and timing QOR of Global placement.

place_global_module_aware_spare

`place_global_module_aware_spare {true | false}`

Default: `false`

Data_type: bool, read/write

Places spare cells in the netlist within a hierarchical module, even if no region or fence constraints are specified.

- When this parameter is `true`, spare cells in a logical hierarchy are placed within the bounds of the hierarchy.
- When this parameter is `false`, spare cells are spread out within the placement area, and are not bound to a hierarchical module.

place_global_module_padding

```
place_global_module_padding {module factor}
```

Default: "" (empty string)

Data_type: string, read/write

Specifies a module in which to add padding (placement clearance) and a factor to use to calculate the padding dimension. Adding padding reduces placement density and localized congestion (hotspots) by spreading out cell instances in the specified modules. Module padding provides guidance for global placement only, and is ignored during placement legalization (`place_detail`).

- `module`: Specifies a hierarchical instance to which the factor applies. You can use "prefix matching" when you specify the instance name. For example, you can use the following command to specify the module padding for all u_ram_block_{2,4,6} module.

  ```
  set_db place_global_module_padding {u_ram_block_ 2.0}
  ```

- `factor`: Specifies a factor to use to calculate the padding dimension. The placer multiplies the instance area of all cell instances under the specified module by the factor. For example, a factor of 2 means that the placer "sees" each cell as twice its actual size. In most cases, a factor of 1.2 (which increases the area by 20 percent) is good enough. The placer ignores factors that are less than 1.0.

place_global_place_io_pins

```
place_global_place_io_pins {true | false}
```

Default: false

Data_type: bool, read/write

Moves placed and unplaced I/O pins, based on the placement of connected instances in an attempt to find a better I/O pin placement than the one specified in the I/O pin placement or floorplan file. At the end of global placement, the I/O pin location is legalized.

When the value of this parameter is `false`, I/O pins are ignored during global placement and no legalization is done at the end.

If `place_design` does not place the standard cells, for example if all instances are fixed or if there is no placeable area, then `place_design` also skips I/O pin assignment.

place_global_sdp_alignment

```
place_global_sdp_alignment {true | false}
```

Default: `false`

Data_type: bool, read/write

Ensures SDP alignment during the global cell placement.

The user-specified SDP alignment is preserve during the global cell placement.

place_global_sdp_place

```
place_global_sdp_place {true | false}
```

Default: `false`

Data_type: bool, read/write

Enables SDP placement for `place_opt_design` command and where placer will place SDP cells closely as a group.

place_global_reorder_scan

```
place_global_reorder_scan {true | false}
```

Default: `true`

Data_type: bool, read/write

Ignores scan chain connectivity during placement and performs scan chain reordering after placement.

If you specify `false` for this parameter, the software does not perform scan chain reordering after placement.

place_global_soft_guide_strength

```
place_global_soft_guide_strength {low | medium | high}
```

Default: `low`

Data_type: enum {low medium high}, read/write

Instructs placement engine to place those modules which have been specified as softguide in the

floorplan file closer. The degree of closeness depends on the strength given.

In case, there are fixed/preplaced instances, the module placed gets affected by those fixed instances and therefore placed close to them.

place_global_timing_effort

```
place_global_timing_effort {medium|high}
```

Default: `medium`

Data_type: enum {medium high}, read/write

Specifies the level of effort for timing driven global placer.

Note: `place_global_timing_effort high` is not recommended for congested designs.

⚠ This option is part of a limited-access feature in this release. It is enabled by a variable specified using the `set_limited_access_feature` command. To use this feature, contact your Cadence representative to explain your usage requirements, and make sure this feature meets your needs before deploying it widely.

place_global_uniform_density

```
place_global_uniform_density {true | false}
```

Default: `false`

Data_type: bool, read/write

Enable even cell distribution for designs with less than 70% utilization.

place_hard_fence

```
place_hard_fence {true | false}
```

Default: `true (place_detail honors the constraints)`

Data_type: bool, read/write

Specifies whether `place_detail` and `check_place` honor or ignores the fence and region placement constraints while legalizing placed instances.

Note: In optimization, the fence and region constraints are disabled by default in non-MSV designs for better timing enclosure, which is controlled by `set_db opt_honor_fences` with the `false` default value.

For the fence and region constraints to be honored by optimization commands, both `set_db opt_honor_fences` and `set_db place_hard_fence` must be set to `true`.

place_opt_post_place_tcl

`place_opt_post_place_tcl` *`tcl_file`*

Default: `""`

Data_type: string, read/write

Sources the tcl script after placement and before running optimization inside `place_opt_design`.

place_opt_run_global_place

`place_opt_run_global_place {none seed full}`

Default: `full`

Data_type: enum {none seed full}, read/write

Changes the Global Placement behavior inside `place_opt_design`.

- `none`: Skips place_design and runs only preCTS Optimization inside `place_opt_design`.
- `seed`: Uses seed placement from Concurrent Placement or other sources and runs incremental standard cell placement, then runs preCTS optimization.
- `full`: Runs full global placement and preCTS optimization.

When `place_design -concurrent_macros` is run, it sets the value of `place_opt_run_global_place` to `seed`.

place_spare_update_timing_graph

`place_spare_update_timing_graph {true | false}`

Default: `true`

Data_type: bool, read/write

Updates timing graph during `place_spare_modules`.

place Category Object Attributes

To view help for any object type attribute across different object types, type the following command:

```
help attributeName
```

For example, for help on all the place_status objects, type this:

```
help place_status*
```

For help on a specific object attribute, use this:

```
help inst place_status_cts
```

The following categories of object attributes are available in the place category:

Object Category Name	Attributes
bump	place_status
inst	place_status place_status_cts place_status_effective
port	place_status

reorder_scan

```
reorder_scan
[-help]
[-add_scan_port_prefix prefix]
[-clock_aware {true | false}]
[-keep_power_domain_ports {true | false}]
[-prefer_horizontal {true | false}]
[-prefer_vertical {true | false}]
[-scan_effort {low | medium | high}]
[-skip_mode {skipNone | skipBuffer}]
```

Reorders the scan cells after running placement. Controls the skipping and removal of buffers and inverters in the scan chain. Use this command after running placement with the ignore scan connection attribute (`place_global_ignore_scan true`), and after specifying the scan chain or loading scan chain information in the DEF or TDF files.

Set attributes for this command with the reorder_scan Category Attributes.

This command automatically detects multiple power domains and minimizes the number of scan chains that pass through them. It respects rectilinear and rectangular power domain boundaries. If a scan chain crosses power domains that operate at the same voltage, the software does not insert level shifters. If the scan chain crosses power domains operating at different voltages, the software inserts the appropriate level shifters.

If scan reorder fails, use the `trace_scan -verbose` command to see where the scan chain stops.

Note: Parameters set by this command override the attribute settings of the reorder_scan category.

Note: This command supports multithreading.

Parameters

`-add_scan_port_prefix` *prefix*	Specifies the prefix to be added to scan hports. *Default:* `""`

`-clock_aware {true | false}`	Specifies whether scan reordering is clock tree aware. • When `true`, scan reordering is clock tree aware, and is clock tree driven. • When `false`, scan reordering is not clock tree aware, and is wirelength driven. *Default:* `false` (The software is not clock tree aware) *Data_type*: bool, optional This parameter can also have the value `0, 1` or any floating point number between `0` and `1`. If this parameter is not specified, or if `0` is specified for this parameter, this indicates: Do not consider clock tree at all. If the value is close to `1`, this indicates: Give greater consideration to clock tree.
`-keep_power_domain_ports {true | false}`	Maintains or corrects the hierarchical ports without deleting or inserting shifters, and reorders the remaining scan chain. To turn this option off, use `reorder_scan -keepPDports false` or `reorder_scan_keep_hport false`. *Default:* `true` *Data_type*: bool, optional **Note:** This parameter works in the MSV flow only.
`-prefer_horizontal {true | false}`	Specifies that the preferred direction for scan reordering connections is horizontal. Reorders the scan chain to snake a sideways "S" configuration if the block is short and wide or the design has vertical routing congestion. *Default:* `false` *Data_type*: bool, optional **Note:** If both `-prefer_vertical` and `-prefer_horizontal` are set to `true`, the software does not honor the setting and treats them both as if they were `false`.

`-prefer_vertical {true | false}`	Specifies that the preferred direction for scan reordering connections is vertical. Reorders the scan chain to snake a vertical "S" configuration if the block is long and narrow or the design has horizontal routing congestion. *Default:* `false` *Data_type*: bool, optional **Note:** If both `-prefer_vertical` and `-prefer_horizontal` are set to `true`, the software does not honor the setting and treats them both as if they were `false`.
`-scan_effort`	Specifies the effort level for reordering scan chains. *Default:* `-medium` *Data_type*: enum, optional Specify one of the following values: • `-high`: Runs more iterations of scan reorder to achieve better net length. Using this parameter increases the run time. • `-low`: Runs fewer iterations of scan reorder to minimize net length quickly. • `-medium`: Runs a normal number of iterations.
`-skip_mode {skipNone | skipBuffer}`	Specifies how scan reordering handles buffers and inverters in the scan chain. *Default:* `skipNone` *Data_type*: enum, optional Specify one of the following: • `-skip_none`: Specifies that buffers and inverters remain after the scan chain reorder. • `-skip_buffer`: Ignores buffers in the scan chain. If a buffer or inverter is not connected elsewhere in the design, `reorder_scan` removes the buffer.

Example

The following command reconnects the scan chain so that buffers are skipped:

```
reorder_scan -skip_mode skipBuffer
```

Related Information

- `reorder_scan Category Attributes`
- `trace_scan`
- Placing the Design chapter in the *Innovus Stylus Common UI User Guide*
  - Optimizing and Reordering Scan Chains
- Place Menu chapter in the *Innovus Stylus Common UI Menu Reference*.

reorder_scan Category Attributes

The root attributes in the reorder_scan category are:

- reorder_scan_add_scan_port_prefix
- reorder_scan_clock_aware
- reorder_scan_comp_logic
- reorder_scan_enable_for_partition
- reorder_scan_keep_hport
- reorder_scan_keep_hinst_port_name
- reorder_scan_prefer_horizontal
- reorder_scan_prefer_vertical
- reorder_scan_effort
- reorder_scan_skip_mode
- reorder_scan_swap_effort

To set the value of any root attribute, use the `set_db` command:

```
set_db attribute_name value
```

To get the current value of any root attribute, use the `get_db` command:

```
get_db attribute_name
```

To get the current value of all attributes in the `reorder_scan` category, use the following command:

```
get_db -category reorder_scan
```

reorder_scan_add_scan_port_prefix

`reorder_scan_add_scan_port_prefix` *prefix*

Default: `""`

Data_type: string, read/write

Specifies the prefix to be added to scan hports.

reorder_scan_clock_aware

`reorder_scan_clock_aware {true|false}`

Default: `false` (The software is not clock tree aware.)

Data_type: bool, read/write

Specifies whether scan reordering is clock tree aware.

- When `true`, scan reordering is clock tree aware, and is clock tree driven.
- When `false`, scan reordering is not clock tree aware, and is wirelength driven.

reorder_scan_comp_logic

`reorder_scan_comp_logic {true|false}`

Default: `false`

Data_type: bool, read/write

If set to `true`, enables `trace_scan` trace through multiple input logic gates (whose input/output pin is not defined in the input scan DEF) in the scan path.

Note: When `-reorder_scan_comp_logic` is set to true, there can be multiple valid scan paths. In that case, `trace_scan` will select one of the valid paths. However, it is recommended that you specify the relevant gate's input/output pin to avoid trace through alternative paths.

Related Command

- `trace_scan`

reorder_scan_enable_for_partition

`reorder_scan_enable_for_partition {true|false}`

Default: `false`

Data_type: bool, read/write

Enables `reorder_scan` for a design with partition.

reorder_scan_keep_hport

```
reorder_scan_keep_hport {true|false}
```

Default: `true`

Data_type: bool, read/write

Maintains or corrects the hierarchical ports without deleting or inserting shifters, and reorders the remaining scan chain.

Note: This parameter works in the MSV flow only.

reorder_scan_keep_hinst_port_name

```
reorder_scan_keep_hinst_port_name filename
```

Default: `""`

Data_type: string, read/write

Specifies a file that lists the names of hierarchical instances whose ports cannot be changed, removed, duplicated, or reordered during scan reordering.

reorder_scan_prefer_horizontal

```
reorder_scan_prefer_horizontal {true|false}
```

Default: `false`

Data_type: bool, read/write

Specifies that the preferred direction for scan reordering connections is horizontal. Reorders the scan chain to snake a sideways "S" configuration if the block is short and wide or the design has vertical routing congestion.

Note: If both `-reorder_scan_prefer_vertical` and `-reorder_scan_prefer_horizontal` are set to `true`, the software does not honor the setting and treats them both as if they were `false`.

reorder_scan_prefer_vertical

```
reorder_scan_prefer_vertical {true|false}
```

Default: `false`

Data_type: bool, read/write

Specifies that the preferred direction for scan reordering connections is vertical. Reorders the scan chain to snake a vertical "S" configuration if the block is long and narrow or the design has horizontal routing congestion.

reorder_scan_effort

```
reorder_scan_effort {low|medium|high|auto}
```

Default: `auto`

Data_type: enum {low medium high auto}, read/write

Specifies the effort level for reordering scan chains.

Specify one of the following values:

- `high`: Runs more iterations of scan reorder to achieve better net length. Using this parameter increases the run time.
- `low`: Runs fewer iterations of scan reorder to minimize net length quickly.
- `medium`: Runs a normal number of iterations.

reorder_scan_skip_mode

```
reorder_scan_skip_mode {skip_none|skip_buffer|skip_floating_buffer}
```

Default: `skipNone`

Data_type: enum {skip_none skip_buffer skip_floating_buffer}, read/write

Specifies how scan reordering handles buffers and inverters in the scan chain.

Specify one of the following:

- `skip_none`: Specifies that buffers and inverters remain after the scan chain reorder.
- `skip_buffer`: Ignores buffers in the scan chain. If a buffer or inverter is not connected elsewhere in the design, `reorder_scan` removes the buffer.
- `skip_floating_buffer`: Skips only buffers in the floating section and keeps the buffers that belong to an ordered section.

  Note: During scan reordering, the order of scan elements in the ordered section must be kept as importing scan specification. However, the order of the floating elements can be changed.

Related Command

- reorder_scan

reorder_scan_swap_effort

```
reorder_scan_swap_effort {low|medium|high}
```

Default: `high`

Data_type: enum {low medium high}, read/write

Specifies the effort level for reorder_scan different chains swapping.

Related Command

- reorder_scan

report_cell_edge_spacing

`report_cell_edge_spacing`

Reports all the cell edge spacing rules in both terminal and log file.

Parameters

None

Related Information

- `delete_cell_edge_spacing`
- `set_cell_edge_spacing`

report_cell_edge_type

```
report_cell_edge_type
[-help]
[-out_file cellName]
[-file fileName]
```

Reports the edge type of the specified cell.

Parameters

`-cell cellName`	Specifies the cell name for which edge type is to be reported. If the cell name is not specified, all the cell edge types are reported in the edge_type.rpt file report file. *Data_type*: string, optional
`-out_file fileName`	Specifies the file name to dump cell edge type(s) . If no file name is specified, the edge_type.rpt file is used to generate the report. *Data_type*: string, optional
`-help`	Outputs a brief description that includes type and default information for each `report_cell_edge_type` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man report_cell_edge_type`.

Related Information

- `set_cell_edge_type`

report_cell_padding

```
report_cell_padding
[-help]
[-cell cellName]
[-out_file fileName]
```

Prints the number of cells, which contain user-specified padding, to the log file. Optionally, you can obtain a complete list of cells with their padding information in a file using the `-file` parameter. The format of this file is same as used in the `set_cell_padding` command, that is, *`cellName padding`* (in grid units).

Parameters

`-cell` *`cellName`*	Specifies the cell name to report padding. *Data_type*: string, optional
`-out_file` *`fileName`*	Specifies the generated file name to dump cell padding. *Data_type*: string, optional
`-help`	Outputs a brief description that includes type and default information for each `report_cell_padding` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man report_cell_padding`.

Examples

The following steps show the usage of `report_cell_padding` command:

1. Specify the following padding using the `set_cell_padding` command:

   ```
   set_cell_padding FF_SCAN 2
   set_cell_padding DFFSHQX1 5
   set_cell_padding DFFSRX4 4
   ```

2. Run the `report_cell_padding` command. After running this command, you get the value `3` and the following message is printed to the log file:

```
3 cells have padding.
```

3. To print the above-mentioned list of cells in the file `myfile`, specify the following command:

```
report_cell_padding -out_file myfile
```

4. After running this command, the following contents of `myfile` are displayed:

```
FF_SCAN 2
DFFSHQX1 5
DFFSRX4 4
# 3 cells have padding.
```

Related Information

- `delete_cell_padding`
- `set_cell_padding`

report_cell_stack_area

```
report_cell_stack_area
[-help]
[-cell cell_name]
[-file file_name]
[-group group_name]
```

Reports the cell stack area definition for cells.

Parameters

`-cell cell_name`	Specifies the cell(s) for which the cell stack area definition needs to be reported. *Data_type*: string, optional
`-file file_name`	Specifies the file name in which the reported information is saved. By default, all cell stack groups are reported to the `cell_stack_area.rpt` file. *Data_type*: string, optional
`-group group_name`	Specifies the group(s) for which the cell stack area definition needs to be reported. *Data_type*: string, optional
`-help`	Outputs a brief description that includes type and default information for each `report_cell_stack_area` parameter. For a detailed description of the command and all of its parameters, use the man command: `man report_cell_stack_area`

Related Information

- `delete_cell_stack_area`
- `set_cell_stack_area`

report_cell_stack_group

```
report_cell_stack_group
[-help]
[-file file_name]
[-name group_name]
```

Reports the cell stack group definition.

Parameters

`-file` *file_name*	Specifies the file name in which the reported information is saved. By default, all cell stack groups are reported to the `cell_stack_group.rpt` file. *Data_type*: string, optional
`-help`	Outputs a brief description that includes type and default information for each `report_cell_stack_group` parameter. For a detailed description of the command and all of its parameters, use the man command: `man report_cell_stack_group`
`-name` *group_name*	Specifies the group(s) that needs to be reported. *Data_type*: string, optional

Related Information

- `delete_cell_stack_group`
- `set_cell_stack_group`
- Place Menu chapter in the *Innovus Stylus Common UI Menu Reference*.

report_cell_virtual_align

```
report_cell_virtual_align
[-help]
[-cells cells]
```

Reports the cells that need to be virtually aligned.

Parameters

`-cells` *cells*	Specifies the lib cells that require virtual alignment. *Data_type*: (base_cell)+, optional
`-help`	Outputs a brief description that includes type and default information for each `report_cell_virtual_align` parameter. For a detailed description of the command and all of its parameters, use the man command: `man report_cell_virtual_align`

Related Information

- `delete_cell_virtual_align`
- `set_cell_virtual_align`

report_congestion

```
report_congestion
[-help]
[-3d]
[-hotspot]
[-include_blockage]
[-num_hotspot number]
[-overflow]
```

Reports the average congestion and the local hotspot score. Route needs be called before using this command.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `report_congestion` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man report_congestion`.
`-3d`	Honors 3d congestion map. When specified, it enables the layer-based congestion reporting.
`-hotspot`	Reports the local hotspot score. It reports the maximum and total hotspot area. Hotspot score refers to the contiguous area of gcells (there is some normalization done internally) which have routing overflow. Typically, a max hotspot score < 100 indicates that the design is routable. • Max hotspot – It is the largest contiguous area of gcells with overflow • Total hotspot – It is the total of all hotspots Greater the number of adjacent gcells with overflow, bigger will be the hotspot. **Note:** You can use this parameter to view the congestion hotspot regions seen by the router after global route. *Data_type*: bool, optional
`-include_blockage`	Includes blockage area in the hotspot report. *Data_type*: bool, optional
`-num_hotspot` *number*	Specifies the number of hotspots to be reported. *Data_type*: int, optional
`-overflow`	Reports horizontal and vertical overflow. *Data_type*: bool, optional

Examples

- The following command reports the congestion:

```
report_congestion -overflow
```

Output:

```
Usage: (10.0%H 8.1%V) = (2.425e+05um 3.107e+05um) = (72854 61452)
Overflow: 210 = 7 (0.01% H) + 202 (0.33% V)

Congestion distribution:

Remain cntH cntV
------------------------------------
-5: 0 0.00% 1 0.00%
-4: 0 0.00% 7 0.01%
-3: 0 0.00% 18 0.03%
-2: 1 0.00% 41 0.07%
-1: 6 0.01% 97 0.16%
------------------------------------
0: 125 0.21% 256 0.42%
1: 568 0.95% 685 1.13%
2: 285 0.47% 7995 13.14%
3: 889 1.48% 6145 10.10%
4: 7457 12.41% 917 1.51%
5: 50749 84.47% 44681 73.44%
```

- The following command reports the local hotspot score:

```
report_congestion -hotspot
```

Output:

```
[hotspot] +-----+-----------------------------------+---------------+
[hotspot] | Top |           hotspot bbox            | hotspot score |
[hotspot] +-----+-----------------------------------+---------------+
[hotspot] |  1  |       32.16 262.56 216.48 400.80  |    54.89      |
[hotspot] +-----+-----------------------------------+---------------+
[hotspot] |  2  |       1.44 446.88 124.32 585.12   |    46.89      |
[hotspot] +-----+-----------------------------------+---------------+
[hotspot] |  3  |      554.40 170.40 638.88 277.92  |    30.89      |
[hotspot] +-----+-----------------------------------+---------------+
[hotspot] |  4  |      477.60 708.00 585.12 800.16  |    25.22      |
[hotspot] +-----+-----------------------------------+---------------+
[hotspot] |  5  |     62.88 108.96 185.76 170.40    |    21.22      |
```

```
[hotspot] +-----+-----------------------------------+---------------+
```

- The following command reports the local hotspot score:

```
report_congestion -hotspot -3d
[hotspot] +------------+--------------+--------------+-----------------------------
--------+
[hotspot] |    layer   |  max hotspot  | total hotspot |            hotspot bbox
        |
[hotspot] +------------+--------------+--------------+-----------------------------
--------+
[hotspot] |    M1(H)   |          0.00 |          0.00 |   (none)
       |
[hotspot] |    M2(V)   |         19.78 |        109.56 |   554.40   600.48   615.84
692.64 |
[hotspot] |    M3(H)   |        174.33 |        693.67 |    47.52   201.12   293.28
723.36 |
[hotspot] |    M4(V)   |          0.89 |          6.67 |   247.20   354.72   277.92
385.44 |
[hotspot] |    M5(H)   |          4.89 |         26.22 |   108.96   170.40   139.68
201.12 |
[hotspot] +------------+--------------+--------------+-----------------------------
--------+
[hotspot] |   worst    | (M3)   174.33 | (M3)   693.67 |
        |
[hotspot] +------------+--------------+--------------+-----------------------------
--------+
[hotspot] | all layers |        168.89 |        654.89 |
        |
[hotspot] +------------+--------------+--------------+-----------------------------
--------+
```

```
Local HotSpot Analysis (3d): normalized congestion max/total hotspot area =
168.89/654.89 (area is in unit of 4 std-cell row bins)
```

Related Information

- place_fix_congestion

report_density_in_area

```
report_density_in_area
-area
```

Outputs placement density for a specific area in the design. Considers placement blockages and honors the metal layers specified by the `place_detail_preroute_as_obs` attribute. The argument type is box in units of microns. This command is optimized for small area interactive queries.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `report_density_in_area` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man report_density_in_area`
`-area`	Specifies box coordinates (*`x1, y1, x2, y2`*) in micron. • *x1:* Specifies the area's lower left point on the x axis. • *y1:* Specifies the area's lower left point on the y axis. • *x2:* Specifies the area's upper right point on the x axis. • *y2:* Specifies the area's upper right point on the y axis. *Data_type*: rect, required

Examples

The following command reports the placement density for the specific area 2835.342 3381.807 4807.673 4530.904:

```
report_density_in_area 2835.342 3381.807 4807.673 4530.904
```

The output of this command is:

```
StdInstArea/freeSpace = 27.0%(197105.76/728824.32)

macroInstArea/totArea = 67.7%(1539687.84/2272665.60)

powerMetalArea/totArea = 0.0%(0.00/2272665.60)
```

```
PlacementObsArea/totArea = 0.0%(0.00/2272665.60)

Utilization in area (2835342 3381807) (4807673 4530904) = 27.04%
```

Related Information

- place Category Attributes

report_density_map

```
report_density_map
[-help]
[-grid_length rowHeight]
[-grid_unit microns]
[-ignore_block {true|false}]
[-ignore_filler {true|false}]
[-threshold density]
```

Generates a map that uses colors to represent placement density and generates a placement density report. If you run this command without parameters, it generates a map with a threshold of 0.75 and measures the grid in row height.

Note: You can also use `report_pin_density Category Attributes` to set the threshold, and specify whether the map displays bins in microns or rows.

Use this command any time after placement.

Use the *Display Density Map* form to select colors and step-size for the density levels in the map.

Parameters

`-grid_unit` *microns*	Specifies the value (in micron) of side of square grid for density calculations. *Data_type*: float, optional
`-grid_length` *rowHeight*	Specifies the value (in row-height) of side of square grid for pin density calculations. *Data_type*: int, optional
`-help`	Outputs a brief description that includes type and default information for each `report_density_map` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man report_density_map`.
`-ignore_block {true|false}`	Skips block cells instances from both supply and demand calculation. *Data_type*: bool, optional
`-ignore_filler`	Ignores the density of fillers in the post-filler density heat map to show the density of standard cells . *Default*: `false` *Data_type*: bool, optional
`-threshold` *density*	Specifies a threshold density value for the grid. Any grid under the threshold value is not reported. The value must be positive. *Data_type*: float, optional

Example

The following are output from command `report_density_map`. It generates a map with default threshold of 0.75 and measured in row height:

```
powerDomain TOP: bins with density > 0.75 = 0.519 % ( 6 / 1156 )

bin size: 59 sites by 10 row(s). total 1156 bins ( 34 by 34 )

density range #bins %

0.75 - 0.8 5 0.4325

0.8 - 0.85 1 0.08651
```

```
total 6 0.519
```

Related Information

- `report_pin_density_map`
- `report_pin_density Category Attributes`
- Place Menu chapter of the *Innovus Stylus Common UI Menu Reference*
    - Display Density Map

report_inst_padding

```
report_inst_padding {inst_name | -all}
```

Reports the padding for a specific instance, or all instance padding in the design. This command can be used if there is instance padding specified.

Parameters

`-all`	Reports all instance padding in the design. *Data_type*: bool, required
`inst_name`	Reports the padding for the specified instance. *Data_type*: string, required

Example

The following command reports the padding for a specified instance named U19567:

```
report_inst_padding U19567
```

It outputs following:

```
Padding for Instant [U19567] is 2.
```

Related Information

- `delete_inst_padding`
- `set_inst_padding`
- Placing the Design chapter in the *Innovus Stylus Common UI User Guide*
  - *Adding Padding*

report_inst_space_group

```
report_inst_space_group
[-help]
[-file fileName]
[-group groupName]
```

Reports all instance space groups into the inst_space_group.rpt file by default.

Parameters

`-file` *fileName*	Specifies the file name to dump space group. *Data_type*: string, optional
`-group` *groupName*	Specifies the name of the group to be reported. *Data_type*: string, optional
`-help`	Outputs a brief description that includes type and default information for each `report_inst_space_group` parameter. For a detailed description of the command and all of its parameters, use the man command: `man report_inst_space_group`.

Related Information

- `create_inst_space_group`
- `delete_inst_space_group`

report_jtag_inst

```
report_jtag_inst
[-out_file]
```

Displays JTAG information specified by the `set_jtag` command or creates an output file that contains the information.

Parameters

`-out_file`	Specifies the name of the file. If you do not specify this parameter, the command prints the information to the screen without saving it in a file. *Data_type*: string, optional

Related Topics

- `set_jtag`
- `set_jtag_to_unplaced`
- Placing the Design chapter in the *Innovus Stylus Common UI User Guide*
  - Specifying and Placing JTAG and Other Cells Close to the I/Os

report_pg_keep_out

```
report_pg_keep_out
[-help]
[-file file_name]
{-cell lib_cell_name | -inst instance_name}
```

Reports all PG keepout boxes in the design if no option is specified.

Parameters

`-cell` *lib_cell_name*	Reports PG keepout boxes applied to the specified lib cell only. *Data_type*: string, optional
`-file` *file_name*	Specifies the file name where all the details are dumped. *Data_type*: string, optional
`-help`	Outputs a brief description that includes type and default information for each report_pg_keep_out parameter. For a detailed description of the command and all of its parameters, use the man command: `man report_pg_keep_out`
`-inst` *instance_name*	Reports PG keepout boxes applied to the specified instance only. *Data_type*: string, optional

Related Information

- `create_pg_keep_out`
- `delete_pg_keep_out`

report_pin_density

```
report_pin_density
[-help]
[-area <x1 y1 x2 y2>]
```

Outputs pin density for the design.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `report_pin_density` parameter. For a detailed description of the command and all of its parameters, use the man command: `man report_pin_density`
`-area`	Species the box coordinates in microns. • *x1*: Specifies the area's lower left point on the x axis. • *y1*: Specifies the area's lower left point on the y axis. • *x2*: Specifies the area's upper right point on the x axis. • *y2*: Specifies the area's upper right point on the y axis. *Data_type*: rect, optional The argument type is box in units of microns.

Examples

The following command reports the pin density for the specific area 93.85, -645.08, 102.83, -640.33:

```
report_pin_density -area 93.85 -645.08 102.83 -640.33
calculate placement pin density in box (93.85, -645.08, 102.83, -640.33)
Pin Density = 0.391304.
= total # of pins 36 / total Instance area 92.00.
```

Related Information

- report_pin_density Category Attributes
- report_pin_density_map

report_pin_density_map

```
report_pin_density_map
[-help]
[-step step_value]
[-grid_unit micron]
[-grid_length numberRow]
[-threshold density]
```

Generates a map that uses colors to represent pin density and generates a pin density report. Use this command any time after placement.

> You can also use report_pin_density Category Attributes to set the threshold and specify whether the map displays bins in microns or rows.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `report_pin_density_map` parameter. For a detailed description of the command and all of its parameters, use the man command: `man report_pin_density_map`
`-step` *step_value*	Specifies the range step for displaying the pin density map. *Default*: `0.05` *Data_type*: float, optional
`-grid_unit` *micron*	Specifies the value of side of square grid for pin density calculations. *Default:* `50` *Minimum*: `0.000000` *Maximum:* `1000.000000` *Data_type*: int, optional
`-grid_length` *numberRow*	Specifies the value (in row-height) of side of square grid for pin density calculations. *Default:* `10` *Minimum:* `1` *Maximum:* `100` *Data_type*: float, optional
`-threshold` *density*	Specifies a threshold density value over which the densities must be reported. Any grid under the threshold value is not reported. The value must be positive. *Default*: `0.5` *Minimum*: `0.000000` *Maximum*: `2.000000` *Data_type*

Examples

The following command reports the pin density map:

```
report_pin_density_map
*** Starting report pin density ***
default core: bins with pin density > 0.5 = 0 % (0 / 1)
bin size: 90 sites by 10 row(s). total 1 bins ( 1 by 1 )
pin density range   #bins     %
total                 0       0
```

Related Information

- report_pin_density
- report_pin_density Category Attributes

report_pin_density Category Attributes

The root attributes in the report_place category are:

- report_pin_density_map_grid_unit
- report_pin_density_map_grid_size
- report_pin_density_map_threshold
- report_pin_density_map_display_step

To set the value of any root attribute, use the `set_db` command:

```
set_db attribute_name value
```

To get the current value of any root attribute, use the `get_db` command:

```
get_db attribute_name
```

To get the current value of all attributes in the `report_pin_density` category, use the following command:

```
get_db -category report_pin_density
```

report_pin_density_map_grid_unit

`report_pin_density_map_grid_unit` *micron*

Default: 50.0

Data_type: double, read/write

Specifies the value (in micron) of side of the square grid for pin density calculations.

report_pin_density_map_grid_size

`report_pin_density_map_grid_size` *numberRow*

Default: `10`

Data_type: int, read/write

Specifies the value (in row-height) of side of the square grid for pin density calculations.

report_pin_density_map_threshold

`report_pin_density_map_threshold` *`density`*

Default: `0.5`

Data_type: double, read/write

Specifies the threshold value over which the densities must be reported.

report_pin_density_map_display_step

`report_pin_density_map_display_step` *`step_value`*

Default: `.05`

Data_type: double, read/write

Specifies the range step for displaying the pin density map calculations.

report_pipelines

```
report_pipelines
[-help]
[-net_groups netGroupNames]
[{-spacing value  | -timing  | -ps_per_um } -out_file fileName]
```

The command can be used for analysis before the flops exist in the netlist. It can also be used post flop insertion to either report expected average wire-length between stages of a pipeline net group, or to report the achieved average distance between the placed stages of the pipeline. When used for analysis, the command reports expected number of PFF stages for each net group based on the user-specified value. Optionally the command generates bus-guides for reported route-path.

Parameters

`-out_file` *fileName*	Specified the output for additional information from the command. The output includes expected placement location of PFF at each stage in analysis flow. *Data_type*: string, optional
`-net_groups` *netGroupNames*	Restricts the report to the net group name specified. If no net group name is specified, all pipeline net groups are reported. *Data_type*: string, optional
`-spacing` *value*	Given the target approximate wire-length spacing between stages, `-spacing` suggests the number of flop stages to insert. *Data_type*: float, optional
`-timing`	`-timing` is used to automatically estimate the number of pipeline FFs needed, based on timing delay on routing layers and the clock cycle requirement. It is mutually exclusive to option `-spacing` because the spacing can be decided automatically. *Data_type*: bool, optional
`-ps_per_um`	`-ps_per_um` is used to automatically find maximum spacing using pspm model. *Data_type*: bool, optional

Example

- The following command reports expected number of PFF stages for net group "`netGroup1`" if the distance between each stage is around 300 microns:

  ```
  report_pipelines -net_groups netGroup1 -spacing 300
  ```

- The following command reports the expected number of PFF stages for net group `netGroup1` if the distance between each stage is around 300 microns. Additional information of expected placement location of each stage are written in the file `PFF.rpt`:

  ```
  report_pipelines -net_groups netGroup1 -spacing 300 -out_file PFF.rpt
  ```

Related Information

- `place_pipelines`
- `delete_pipeline_net_group`

report_place_density Category Attributes

The root attributes in the report_place category are:

- report_place_density_map_grid_size
- report_place_density_map_grid_unit
- report_place_density_map_ignore_block
- report_place_density_map_ignore_filler
- report_place_density_map_threshold

To set the value of any root attribute, use the `set_db` command:

```
set_db attribute_name value
```

To get the current value of any root attribute, use the `get_db` command:

```
get_db attribute_name
```

To get the current value of all attributes in the `report_place_density` category, use the following command:

```
get_db -category report_place_density
```

report_place_density_map_grid_size

`report_place_density_map_grid_size` *numberRows*

Default: 10

Data_type: int, read/write

Specifies the horizontal or vertical dimension of a grid in the map, measured in number of rows. If you specify a value less than `1`, the software uses 1.

report_place_density_map_grid_unit

`report_place_density_map_grid_unit` *microns*

Default: `50.0`

Data_type: double, read/write

Specifies the horizontal or vertical dimension of a grid in the map, measured in microns.

report_place_density_map_ignore_block

`report_place_density_map_ignore_block {true | false}`

Default: `false`

Data_type: bool, read/write

Skips block cells instances from both supply and demand calculation.

report_place_density_map_ignore_filler

`report_place_density_map_ignore_filler {true | false}`

Default: `false`

Data_type: bool, read/write

Ignores the density of fillers in the post-filler density heat map to show the density of standard cells.

report_place_density_map_threshold

`report_place_density_map_threshold` *`density`*

Default: `0.75`

Data_type: double, read/write

Specifies a threshold density value for the grid. Any grid under the threshold value is not reported. The value must be positive. Cadence recommends an upper limit of `1.20`.

Related Command

- `report_density_map`

report_scan_cells

`report_scan_cells`

Displays a list of the scan cells that are read from the timing library or by the `set_scan_cell` command. This command can be called in any stage of flow after importing design if there are scan cell specifications in library; or else it can be used after the `set_scan_cell` command.

Parameters

None

Example

Following is an example of the output from `report_scan_cells`:

```
ScanFF SEDFFXL
  FT SI is test_scan_in
  FT SE is Scan Enable Clock
  FT QN is test_scan_out_inverted
  FT Q is test_scan_out
ScanFF SEDFFX4
  FT SI is test_scan_in
  FT SE is Scan Enable Clock
  FT QN is test_scan_out_inverted
  FT Q is test_scan_out
...
```

Related Information

- `delete_scan_cell`
- `set_scan_cell`
- Placing the Design chapter in the *Innovus Stylus Common UI User Guide*
  - Optimizing and Reordering Scan Chains

report_scan_chain

```
report_scan_chain
[-help]
[-chain chainName]
[-out_file fileName]
[-verbose {true|false}]
```

Prints the scan chain information.

Parameters

`-chain` *chainName*	Specifies the scan chain name. *Data_type*: string, optional
`-out_file` *fileName*	Specifies the output file name of scan chain. *Data_type*: string, optional
`-help`	Outputs a brief description that includes type and default information for each `report_scan_chain` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man report_scan_chain`.
`-verbose {true|false}`	Prints out more message. *Data_type*: bool, optional

Related Information

- `delete_scan_chain`
- `create_scan_chain`

report_scan_chain_partitions

```
report_scan_chain_partitions
[-help]
```

Generates a report that lists compatible scan chains in scan chain partitions.

Specify scan chain partitions with the following command:

`create_scan_chain_partition`

Parameters

None

Example

To see a list of the compatible scan chain groups in a design with one partition, `clk1Domain`, that has two compatible scan chains, type the following command:

```
report_scan_chain_partitions
```

The command outputs the following messages:

```
Info: Scan Chain Partition Group set to:

      Partition group: clk1Domain

                chain: chain1

                chain: chain2
```

Related Topics

- Placing the Design chapter in the *Innovus Stylus Common UI User Guide*
    - Optimizing and Reordering Scan Chains

reset_jtag

```
reset_jtag
{-cell leaf_cellName
 | -inst instanceName
 | -hinst hierarchicalInstanceName
 | -group hierarchicalInstanceName | cellName}
```

Unspecifies instances that were specified by the `set_jtag` command. This command can be used if there is instance specified by the `set_jtag` command.

Parameters

`-cell leaf_cellName`	Unmarks all instances of a specified cell type as JTAG instances. *Data_type*: string, optional
`-group hierarchicalInstanceName | cellName`	Unmarks all hierarchical instances or intermediate instances as JTAG instances. *Data_type*: string, optional
`-hinst hierarchicalInstanceName`	Unspecifies a hierarchical JTAG instance. *Data_type*: string, optional
`-inst instanceName`	Unspecifies a JTAG instance. You can use wildcards (`*` or `?`) with this parameter. *Data_type*: string, optional

Example

- The following command unspecifies cell `jtag_FlipFlop` that was specified by the `set_jtag` command:

  ```
  reset_jtag -cell jtag_FlipFlop
  ```

- The following command unspecifies all instance names that begin with `instG` that were specified by the `set_jtag` command:

  ```
  reset_jtag -inst instG*
  ```

Related Information

- `place_jtag`
- `set_jtag`
- `trace_jtag`
- `report_jtag_inst`
- Placing the Design chapter in the *Innovus Stylus Common UI User Guide*

reset_selective_blockage_gate

```
reset_selective_blockage_gate
[-help]
[-cells cellName]
[-insts instName]
```

Specifies cells and instances that are forbidden to be placed inside the soft blockage during legalization.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `reset_selective_blockage_gate` parameter. For a detailed description of the command and all of its parameters, use the man command: `man reset_selective_blockage_gate`.
`-cells cellName`	Specifies the cell type name as illegal for soft blockage. You can use wildcards (`*` or `?`) with this parameter. *Data_type*: string, optional
`-insts` *`instName`*	Specifies the instance name as illegal for soft blockage. You can use wildcards (`*` or `?`) with this parameter. *Data_type*: string, optional

Examples

The following command specifies that cell `NR2RLX1` and all instance names that begin with `MEMCTLU` cannot be moved into soft blockage during legalization.

```
reset_selective_blockage_gate -cells NR2RLX1 -insts MEMCTLU*
```

Related Information

- `set_selective_blockage_gate`

reset_spare_insts

```
reset_spare_insts
[-help]
{-cell cellName | -inst instName | -hinst hinstName}
```

Asserts the specified instance is not a spare gate. This command can be used after `set_spare_insts`.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `reset_spare_insts` parameter. For a detailed description of the command and all of its parameters, use the man command: `man reset_spare_insts`.
`-cell` *cellName*	Specifies the name of the spare cell to be reset. Wildcards are supported. *Data_type*: string, optional
`-hinst` *hinstName*	Specifies the name of the spare hierarchical instance to be reset. Wildcards are supported. *Data_type*: string, optional
`-inst` *instName*	Specifies the name of the spare instance to be reset. Wildcards are supported. *Data_type*: string, optional

Examples

- In the following example, “AND2X1” cell is defined as a spare cell in the design. The example shows the query result of the spare cell of AND2X1:

```
get_db insts -if {.is_spare == true && .base_cell.name == AND2X1}

inst:DTMF_CHIP/spr_484/spr_gate0 inst:DTMF_CHIP/spr_483/spr_gate0
inst:DTMF_CHIP/spr_482/spr_gate0 inst:DTMF_CHIP/spr_481/spr_gate0
inst:DTMF_CHIP/spr_480/spr_gate0 inst:DTMF_CHIP/spr_479/spr_gate0
inst:DTMF_CHIP/spr_477/spr_gate0
```

- The following example resets the spare cell instance, `spr_477`, with the `-inst` option. As a result, the spare cell is removed from spare module list and is not a spare gate any more:

```
@innovus 15> get_db inst:DTMF_CHIP/spr_477/spr_gate0 .is_spare
true
@innovus 18> reset_spare_insts -inst spr_477/spr_gate0
Resetting spare instance [spr_477/spr_gate0].
Reset 1 spare instances.
@innovus 19> get_db inst:DTMF_CHIP/spr_477/spr_gate0 .is_spare
false
```

- The following commands reset the spare cell hierarchical instance, `spr_479`, with the `-hinst` option, removes it from the spare module list, and it is not a spare gate any more.

```
@innovus 21> get_db inst:DTMF_CHIP/spr_479/spr_gate0 .is_spare
true
@innovus 23> reset_spare_insts -hinst spr_479
Resetting spare mode HInst [spr_479].
@innovus 24> get_db inst:DTMF_CHIP/spr_479/spr_gate0 .is_spare
```

- The following commands reset the spare cell type, `AND2X1`, using the `-cell` option. All the instances of the `AND2X1` cell type are removed from spare module list and it is not a spare gate any more.

```
@innovus 25> reset_spare_insts -cell AND2X1
Resetting all spare instances of cell [AND2X1].
@innovus 26> get_db insts -if {.is_spare == true && .base_cell.name == AND2X1}
```

Related Information

- `set_spare_insts`
- Place Menu chapter of *Innovus Stylus Common UI Menu Reference*.
- Placing the Design chapter of *Innovus Stylus Common UI User Guide*.

set_cell_edge_spacing

```
set_cell_edge_spacing
[-help]
{edgeType1 edgeType2
-spacing spacing
[-except_abutted | -forbidden_spacing ][-except_non_filler_in_between] } | [-reset]
```

Creates the cell edge spacing rule as in `CELLEDGESPACINGTABLE` in LEF.

⚠ The tool by default supports the latest Tcl settings. If a user tries to change the edge spacing, the tool will generate a warning message that resetting edge spacing would overwrite the old value.

Note: This command is not supported for save and restore design.

Parameters

`-except_abutted`	Specifies that two abutted macros of the given edge types are also allowed. *Data_type*: bool, optional
`edgeType1`	Specifies edge `edgeType1` for edge spacing. *Data_type*: string, required
`edgeType2`	Specifies edge `edgeType2` for edge spacing. *Data_type*: string, required
`-except_non_filler_in_between`	Specifies that the edge spacing is only for adjacent instances. If two MACROs (cells) have a edge-spacing rule, say 0.8 micron, then normally the rule applies regardless of presence of another macro (cell) in the gap between. When the `-except_non_filler_in_between` option is used, the rule is exempted when a non-filler cell is present between the two constrained cell edges. *Data_type*: bool, optional
`-forbidden_spacing`	Forbids `edgeType1` and `edgeType2` without the space.
`-help`	Outputs a brief description that includes type and default information for each `set_cell_edge_spacing` parameter. For a detailed description of the command and all of its parameters, use the man command: `man set_cell_edge_spacing`
`-reset`	Resets the cell edge spacing to LEF. *Data_type*: bool, optional
`-spacing` *`spacing`*	Specifies edge spacing between *`edgeType1`* and *`edgeType2`*. If there is no spacing defined between two edge types, it indicates that there is no spacing constraint (or 0.0 µm) between them. *Data_type*: float, required

Example

The following command creates `0.18 µm` spacing between `grp1` and `grp2`:

```
set_cell_edge_spacing grp1 grp2 0.18
```

Related Information

- `delete_cell_edge_spacing`
- `report_cell_edge_spacing`

set_cell_edge_type

```
set_cell_edge_type
[-help]
-cell cellName
{-left edgeType | -right edgeType | -bottom edgeType | -reset}
[{-left edgeType | -right edgeType} [-cell_row rowNumber]]
[{-left edgeType | -right edgeType} [-half {top | bottom}]]
[{-top edgeType | -bottom edgeType} [-range {x y}]] [{-cell cellName} [-reset ]]
```

Specifies edge type for cell edges as `EDGETYPE` in macro LEF. It allows to specify one edge at a time for a particular cell.

To ensure the correct edge type specification, edge types asserted by `set_cell_edge_type` command should be pre-defined by the `set_cell_edge_spacing` command.

For example:

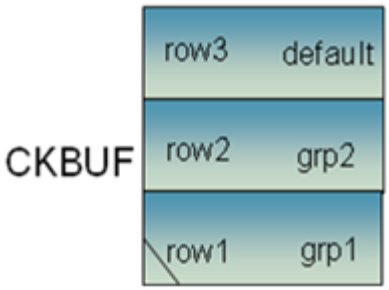
```
set_cell_edge_spacing grp1 grp2 0.18
set_cell_edge_type -cell BUF1X -left grp1
```

If the edge type is specified without corresponding spacing rule, tool gives a warning and ignores the edge type specification.

Note: This command is not supported for save and restore design.

Parameters

`-bottom` *edgeType*	Specifies an edge type on the bottom side. *Data_type*: string, optional
`-cell` *cellName*	Specifies cell to which to add edge constraint. *Data_type*: string, optional

`-cell_row` *rowNumber*	Specifies the row number for horizontal edge type.
	You can define different edge types for each row of multiple height cells using `-cell_row`.
	For example, there is triple height cell CKBUF.
	 
	The following commands define different edge types for different rows of the CKBUF cell.
	`set_cell_edge_type -cell CKBUF -right grp1 -cell_row 1`
	`set_cell_edge_type -cell CKBUF -right grp2 -cell_row 2`
	In the above example, grp1 is applied to the first row (bottom row) of the right edge, grp2 is applied to the second row of right edge, and no specific edge type is defined for the third row of right edge and is of DEFAULT type.
	The legal value of `-cell_row` is from 1 to n, where n is the number of cell height of the cell.
	Data_type: int, optional

`-help`	Outputs a brief description that includes type and default information for each `set_cell_edge_type` parameter. For a detailed description of the command and all of its parameters, use the man command: `man set_cell_edge_type`
`-half {top bottom}`	Defines the edge types for the top or bottom half of the specified row of the cell The following example associates the edge type 'grp1' with the bottom row, top half of the multi-height cell SDFF8: `set_cell_edge_type -cell SDFF8 -right grp1 -cell_row 1 -half top`

`-left` *edgeType*	Specifies an edge type on the left side. *Data_type*: string, optional
`-range {x y}`	Specifies the range for the vertical edge type. *Data_type*: point, optional
`-reset`	Resets the specified cell constraint to lef status. *Data_type*: bool, optional
`-right` *edgeType*	Specifies an edge type on the right side. *Data_type*: string, optional
`-top` *edgeType*	Specifies an edge type on the top side. *Data_type*: string, optional

Examples

- The following commands specify one edge at a time for the `BUF1X` cell

```
set_cell_edge_type -cell BUF1X -left grp1

set_cell_edge_type -cell BUF1X -right grp1 -cell_row 1

set_cell_edge_type -cell BUF1X -right grp1 -cell_row 2
```

- The following commands define different edge types for the first row(bottom row) , right edge of the multi-height cell SDFF8.

```
set_cell_edge_type -cell SDFF8 -right grp1 -cell_row 1 -half top

set_cell_edge_type -cell SDFF8 -right grp2 -cell_row 1 -half bottom
```

  In the above example, grp1 is applied to the first row (bottom row) of the right top-half edge, and grp2 is applied to the first row (bottom row) of right-bottom edge.

Related Information

- delete_base_cell_edge_type
- report_cell_edge_type
- set_cell_edge_spacing

set_cell_padding

```
set_cell_padding
-cells leaf_cellName
{-padding padding | [[-right_side padding] [-left_side padding] [-top_side padding] [-
bottom_side padding]]}
```

Specifies leaf cells to which to add -right_side (placement clearance) and a factor to use to calculate the -right_side dimension. The placer adds the -right_side on the right sides of the specified cells during placement. The -right_side is retained during optimization and CTS.

Use the `get_db` command to query.

If there are congestion problems, delete the -right_side by using the `delete_all_cell_padding` command. For information, see `delete_all_cell_padding -right_side`.

Parameters

`-bottom_side` *padding*	Specifies a factor to calculate the padding dimension on the bottom side. *Minimum*: `0` *Maximum*: `500` *Data_type*: int, optional
-cells leaf_cellName	Specifies the cell name(s) of a cell type to which the factor applies. You can use wildcards (`*`) with this parameter. *Data_type*: string, required
`-left_side` *padding*	Specifies a factor to calculate the padding dimension on the left side. *Data_type*: int, optional
`-padding` *padding*	Specifies a factor to use to calculate the padding dimension. The placer multiplies the placement SITE by the factor. It reads the placement SITE value from the technology file. The factor should be integer number only. The unit of padding on left/right side is the number of sites. The unit of padding on top/bottom side is the number of rows. *Data_type*: int, required
-right_side padding	Specifies a factor to use to calculate the -right_side dimension. The placer multiplies the placement SITE by the factor. It reads the placement SITE value from the technology file. The factor should be integer number only. *Data_type*: int, optional
`-top_side` *padding*	Specifies a factor to calculate the padding dimension on the top side. *Minimum*: `0` *Maximum*: `500` *Data_type*: int, optional

Example

The following command adds -right_side that is twice the placement SITE to leaf cells named `FJK2`:

```
set_cell_padding -right_side FJK2 2
```

Related Information

- `delete_all_cell_padding`
- `report_cell_padding`
- Placing the Design chapter in the *Innovus Stylus Common UI User Guide*
- Place Menu chapter in the *Innovus Stylus Common UI Menu Reference*

set_cell_stack_area

```
set_cell_stack_area
[-help]
-cell cell_name
-group group_name
-rect {x1 y1 x2 y2}
```

Defines the cell area that needs to be involved while calculating stack rule of the corresponding group.

Parameters

`-cell` *cell_name*	Specifies the cell name that has the stack constraint. *Data_type*: string, required
`-help`	Outputs a brief description that includes type and default information for each `set_cell_stack_area` parameter. For a detailed description of the command and all of its parameters, use the man command: man `set_cell_stack_area`
`-group` *group_name*	Specifies the cell group name with the stack constraint. **Note**: The group name must be pre-defined in the LEF file or by the `set_cell_stack_group` command. *Data_type*: string, required
`-rect {x1 y1 x2 y2}`	Specifies the constraint rectangular area with respect to the cell origin. If the whole cell is constrained, the rectangular coordinates match the cell boundary. If the rectilinear area in cell is constrained, divide the rectilinear area into multiple rectangular area then define each rectangular area in separate command lines. *Data_type*: rect, required

Related Information

- delete_cell_stack_area
- report_cell_stack_area
- set_cell_stack_group

set_cell_stack_group

```
set_cell_stack_group
[-help]
-name group_name
{-max_horizontal_length max_continuous_length }
```

Defines the stack rule of a cell group.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `set_cell_stack_group` parameter. For a detailed description of the command and all of its parameters, use the man command: `man set_cell_stack_group`
`-max_horizontal_length` *`max_continuous_length`*	Specifies the maximum continuous length in horizontal direction. *Data_type*: float, required
`-name` *`group_name`*	Specifies the cell group name with the stack constraint. *Data_type*: string, required

Related Information

- `delete_cell_stack_group`
- `report_cell_stack_group`

set_cell_virtual_align

```
set_cell_virtual_align
[-help]
[-cells cells]
[-mask <id>]
[-pin_x_location location_in_microns]
```

Aligns a filler cell that does not have any pin or obs geometries to align, but the cell still needs to be aligned. For example, a GA-Filler cell after the post-mask ECO becomes a logic cell and needs to be correctly aligned for the router to eco-route it.

The `set_cell_virtual_align` command can be used to provide mask for the first track inside the cell, or 'offset' to the first track (or both). You need to specify `-mask` or `-pin_x_location`.

The command auto-detects the layer for vertical alignment. If as per the placement initialization there is no need for vertical alignment (site-width and vertical-track-pitch are same, pin-mask are not fixed), the virtual alignment is turned off. this means that the `set_cell_virtual_align` command will not help.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `set_cell_virtual_align` parameter. For a detailed description of the command and all of its parameters, use the man command: `man set_cell_virtual_align`
`-cells` *cells*	Specifies the lib cells that require virtual alignment. *Data_type*: (base_cell)+, required
`-mask` *virtual_mask_value*	Specifies the virtual mask value for the first track in the cell. If you specify `-mask` only, the offset of the first track in the core is used as the offset. *Data_type*: int, optional
`-pin_x_location` *location_in_microns*	Specifies the location of the virtual geometry from the cell boundary. If you specify offset only, the track can be of any mask-color, as long as offset matches. *Default*: First track location *Data_type*: float, optional

Related Information

- `delete_cell_virtual_align`
- `report_cell_virtual_align`

set_inst_padding

```
set_inst_padding
-inst instName
{-padding padding | [[-right_side padding] [-left_side padding] [-top_side padding] [-
bottom_side padding]]}
```

Specifies a value (number of sites) for the right side for an instance. Padding expands the area of an instance, so that the software leaves extra space around the instance during global placement (`place_design`), which helps reduce congestion in both the horizontal and vertical directions. The `place_detail` and `check_place` commands ignore instance -right_side.

Adding instance -right_side reduces horizontal and vertical congestion, but also decreases utilization.

Note: To remove instance -right_side, use the `delete_inst_padding` command.

Parameters

`-bottom_side` *padding*	Specifies the amount at the bottom, in number of rows. *Data_type*: uint, optional
`-inst` *instName*	Specifies the instance to pad. You can use wildcards (`*`) when you specify the instance. *Data_type*: string, required
`-left_side` *padding*	Specifies the amount of padding on the left side, in number of sites. *Data_type*: uint, optional
`-padding` *padding*	Specifies the amount of padding to add, in the number of sites. *Data_type*: uint, optional
`-right_side` *padding*	Specifies the amount of padding on the right side, in number of sites. *Data_type*: uint, optional
`-top_side` *padding*	Specifies the amount at the top, in number of rows. *Data_type*: uint, optional

Example

The following command adds -right_side of 2 sites around instance named U27316:

```
set_inst_padding -right_side U27316 2
```

It outputs following:

```
Padding for Instant [U27316] is 2.
```

The following command adds -right_side of 3 sites for all instances with prefix U2:

```
set_inst_padding -right_side U2* 3
```

It outputs following:

```
Specified Instant -right_side for [U27313] to 3.

Specified Instant -right_side for [U27314] to 3.

Specified Instant -right_side for [U27315] to 3.

Specified Instant -right_side for [U27316] to 3.

Added -right_side for 4 insts.
```

Related Information

- check_place
- delete_inst_padding
- place_design
- place_detail
- report_inst_padding
- Placing the Design chapter in the *Innovus Stylus Common UI User Guide*

set_jtag

```
set_jtag
{-help}
{-cell leafCellName
    | -inst instName
    | -inst_group
    | -hinst hInstName
    | -group {hInstName | cellName}
    | -group_within_area {hInstName | cellName width height} }
```

Specifies instances to place with the `place_jtag` command. This command should be used before the `place_jtag` command.

Parameters

`-cell` *`leafCellName`*	Marks all instances of the specified cell type as JTAG instances. *Data_type*: string, optional
`-group` *`hInstName	cellName`*	Marks all hierarchical instances or intermediate instances as JTAG instances. *Data_type*: string, optional
`-group_within_area` *`hInstName	cellName width height`*	Marks all instances inside the hierarchical instance or intermediate cells as JTAG instances, and specifies a rectangular width and height where all instances or cells are placed within that rectangular region. *Data_type*: string, optional
-help	Outputs a brief description that includes type and default information for each `set_jtag` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man set_jtag`.
`-hinst` *`hInstName`*	Marks a hierarchical instance as a JTAG instance. *Data_type*: string, optional
`-inst` *`instName`*	Marks an instance as a JTAG instance. You can use a wildcard (`*`) character for this parameter. *Data_type*: string, optional
`-inst_group`	Marks the instGroup name to be made JTAG instance. *Data_type*: string, optional

Example

- The following command specifies that the cell `jtag_FlipFlop` will be placed when using the `place_jtag` command:

  ```
  set_jtag -cell jtag_FlipFlop
  ```

- The following command specifies that all instance names that begin with `instG` will be placed when using the `place_jtag` command:

```
set_jtag -inst instG*
```

Related Topics

- `place_jtag`
- `reset_jtag`
- `set_jtag_to_unplaced`
- `trace_jtag`
- Placing the Design chapter in the *Innovus Stylus Common UI User Guide*
  - Specifying and Placing JTAG and Other Cells Close to the I/Os
- Place Menu chapter in the *Innovus Stylus Common UI Menu Reference*.

set_jtag_to_unplaced

`set_jtag_to_unplaced`

Changes the placement status of JTAG inst from FIXED to PLACED. This command can be used after the `place_jtag` command.

Parameters

None

Examples

The following set of commands places all Jtag instances of cell INVX1 as FIXED, and then using `set_jtag_to_unplaced` to change their status to PLACED:

```
set_jtag -cell INVX1
place_jtag
set_jtag_to_unplaced
```

Related Topics

- `place_jtag`
- `set_jtag`
- Placing the Design chapter in the *Innovus Stylus Common UI User Guide*
    - Specifying and Placing JTAG and Other Cells Close to the I/Os

set_lockup_element

```
set_lockup_element
{-cell leafCellName | -inst instName}
[-scan_in ftname]
[-scan_out ftname]
```

Specifies a cell or instance as a lockup element. Lockup elements are present in scan chains where the chain crosses between two different clock domains or clock phases. Use this command after running placement.

Parameters

`-cell` *leafCellName*	Specifies the cell to be a lockup element. *Data_type*: string, required
`-scan_in` *ftname*	Specifies the name of the scan input. *Data_type*: string, optional
`-inst` *instName*	Specifies the instance to be a lockup element. *Data_type*: string, required
`-scan_out` *ftname*	Specifies the name of the scan output. *Data_type*: string, optional

Example

The following command specifies that cell `TLATX1` is a lockup element with a scan input pin `D` and an output pin `Q`:

```
set_lockup_element -cell TLATX1 -scan_in D -scan_out Q
```

Related Information

- `trace_scan`
- Placing the Design chapter in the *Innovus Stylus Common UI User Guide*
    - Optimizing and Reordering Scan Chains

set_scan_cell

```
set_scan_cell
[-help]
[scan_cell]
[-scan_in ftname]
[-scan_out ftname]
[-scan_clock ftname]
[-scan_enable ftname]
```

Specifies scan cells that are not listed in the timing library.

Parameters

scan_cell	Specifies name of the scan cell. *Data_type*: string, required
-scan_in *ftname*	Specifies the name of the scan input. *Data_type*: string, optional
-scan_out *ftname*	Specifies the name of the scan output. *Data_type*: string, optional
-scan_clock *ftname*	Specifies the name of the scan clock. *Data_type*: string, optional
-scan_enable *ftname*	Specifies name of the scan enable f-term or pin names. *Data_type*: string, optional

Example

The following command specifies cell `SDFFQ` to be a scan cell:

```
set_scan_cell SDFFQ -scan_in SI -scan_out Q
```

Related Information

- `delete_scan_cell`
- `delete_all_scan_cells`
- `place_jtag`
- `report_scan_cells`
- Placing the Design chapter in the *Innovus Stylus Common UI User Guide*
  - Optimizing and Reordering Scan Chains

set_selective_blockage_gate

```
set_selective_blockage_gate
[-help]
[-cells cellName]
[-insts instName]
```

Specifies cells and instances that are allowed to be placed inside the soft blockage during legalization.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `set_selective_blockage_gate` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man set_selective_blockage_gate`.
`-cells cellName`	Specifies the cell type name as legal for soft blockage. You can use wildcards (`*` or `?`) with this parameter. *Data_type*: string, optional
`-insts instName`	Specifies the instance name as legal for soft blockage. You can use wildcards (`*` or `?`) with this parameter. *Data_type*: string, optional

Examples

The following command specifies that cell `NR2RLX1` and all instance names that begin with `MEMCTLU` are allowed to be moved into soft blockage when legalization. `s`

```
et_selective_blockage_gate -cells NR2RLX1 -insts MEMCTLU*
```

Related Information

- `reset_selective_blockage_gate`

set_spare_insts

```
set_spare_insts
{-help}
{-cell leafCellName
   |-insts instanceName
   | -hinst {hierarchicalInstanceName | groupName}}
```

Specifies spare gates to place. To localize spare gate placement, guide the module containing the spare gate during the floorplanning session. This command should be used before placing the design.

Parameters

`-cell` *leafCellName*	Specifies the cell type names of the spare gate cells. You can use this parameter to specify leaf cells (standard cells)--you cannot use it to specify spare modules. You can use wildcards with this parameter. *Data_type*: string, optional
`-help`	Outputs a brief description that includes type and default information for each `set_spare_insts` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man set_spare_insts`.
`-hinst` {*hierarchicalInstanceName* | *groupName*}	Specifies a hierarchical instance or group that contains spare gate cells. A hierarchical instance acts as one unit (a submodule). The units are evenly spaced and are close together when they are distributed. This command does not recognize wildcards in hierarchical instance or group names. *Data_type*: string, optional
`-insts` *instanceName*	Specifies the leaf instance names of the spare gates. You can use wildcards when specifying instance names. *Data_type*: string, optional

Examples

- The following command specifies all cell types `YFD2S_SPG` to be spare gates:

  ```
  set_spare_insts -cell YFD2S_SPG
  ```

- The following command specifies instance `SH16/I1022` to be a spare gate:

  ```
  set_spare_insts -insts SH16/I1022
  ```

- The following command specifies that all instances in `SH16` to be spare gates:

  ```
  set_spare_insts -insts SH16/*
  ```

- The following command treats all instances with the cell type `NAND2X1` as spare cells. If the design uses some of these cells as spares and some as not spare, use the `-insts` parameter instead.

  ```
  set_spare_insts -cell NAND2X1
  ```

Related Information

- `reset_spare_insts`
- Placing the Design chapter in the *Innovus Stylus Common UI User Guide*
  - Placing Spare Cells and Spare Modules
- Place Menu chapter of *Innovus Stylus Common UI Menu Reference*.

swap_well_taps

```
swap_well_taps
[-help]
[-both_violation cellName]
-cells cellNameList
[-check_only]
[-default_cell cellName]
-diffusion_forbidden_spacing float
[-left_violation cellName]
[-right_violation cellName]
[-skip_cells cellNameList]
[-swap_report reportName]
[-violation_report reportName]
```

Swaps to custom tap cell to avoid OD spacing rule.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `swap_well_taps` parameter. For a detailed description of the command and all of its parameters, use the man command: `man swap_well_taps`.
`-both_violation cellName`	Specifies well taps to swap if only violating rules on both sides. *Data_type*: string, optional
`-cells cellNameList`	Specifies well taps to swap if there is violation of rules. *Data_type*: string, optional
`-check_only`	Checks violations only without swapping tap cells. *Data_type*: bool, optional
`-default_cell cellName`	Specifies to swap to default well tap first. This must be one of cells. *Data_type*: string, optional

`-diffusion_forbidden_spacing` *float*	Specifies forbidden spacing rule. *Data_type*: float, required
`-left_violation` *cellName*	Specifies well taps to swap if only violating rules on the left side. *Data_type*: string, optional
`-right_violation` *cellName*	Specifies well taps to swap if only violating rules on the right side. *Data_type*: string, optional
`-skip_cells` *cellNameList*	Specifies the cells that would skip the rule check. *Data_type*: string, optional
`-swap_report` *reportName*	Reports the swapped instances. *Data_type*: string, optional
`-violation_report` *reportName*	Reports the violated instances. *Data_type*: string, optional

Related Information

- `add_well_taps`
- `add_well_taps Category Attributes`

trace_jtag

```
trace_jtag
-in_file jtagPinFile
-out_file jtagInstanceFile
```

Traces JTAG logic between the I/O pins and the pins of given leaf instances. The input file contains a list of pins of instances that are the endpoints of JTAG logic. The command writes out a Tcl file that lists all the JTAG logic traced. You can then source the file to add this information to the Innovus database.

Parameters

`-in_file` *jtagPinFile*	Specifies a file that lists the endpoints of JTAG logic. In the file, list each leaf instance or list the cell and its port. *Data_type*: string, required • To list the leaf instances, use the `instpin` statement, as follows: `instpin` *leafInstanceNamepinName* • If all instances of a cell are used only in JTAG logic, you can list that cell and its port by using the `cellport` statement, as follows: `cellport` *cellTypeNameportName* To avoid tracing non-JTAG cells, the cells you specify in a `cellport` statement must be JTAG-only cells. The cell type cannot be an intermediate cell. It must have at least one physical leaf instance.
`-out_file` *jtagInstanceFile*	Specifies the JTAG logic file in Tcl format. *Data_type*: string, required

Example

The following commands trace the JTAG logic based on the input file `jtag.leaf`, write the output file `jtag.out`, and add the logic specified in the file to the Innovus database:

```
trace_jtag -in_file jtag.leaf -out_file jtag.out
source jtag.out
```

Following is an example of an output file:

```
set_jtag -inst regA
set_jtag -inst regB
set_jtag -inst regC
set_jtag -inst regD
set_jtag -inst regE
set_jtag -inst regF
set_jtag -inst regG
set_jtag -inst regH
```

Related Topics

- set_jtag
- set_jtag_to_unplaced
- Placing the Design chapter in the *Innovus Stylus Common UI User Guide*
  - Specifying and Placing JTAG and Other Cells Close to the I/Os
- Place Menu chapter in the *Innovus Stylus Common UI Menu Reference*.

trace_scan

```
trace_scan
[-help]
[-comp_logic | -no_comp_logic]
[-lockup | -no_lockup]
[-verbose]
```

Records the total number of elements in the scan chain in the log file, and records the starting and ending scan points if the `-verbose` option is used. The `reorder_scan` command automatically calls `trace_scan` internally if you have not previously run `trace_scan`. The `trace_scan` command is used after specifying scan chains using the `create_scan_chain` command.

Note: If scan chains are specified by reading in a DEF file, the software does a native scan trace. The scan DEF file is stored in the database and, when the `trace_scan` command runs, the software matches the scans and honors the ordered sections.

Parameters

`-comp_logic`	Traces through multiple input gate on the scan path.
`-no_comp_logic`	Does not trace through multiple input gate on the scan path.
`-lockup`	Specifies that the tracing detects lockup latches automatically. **Note:** If the lockup elements are flops (non scan flip-flops), the lockup elements must be identified using the `set_lockup_element` command prior to running `trace_scan` in its default mode.
`-no_lockup`	Specifies that the tracing will not detect lockup elements. This is the default.
`-verbose`	Prints the information stored in the logv file, i.e starting and ending scan points, and the cell instance names that `trace_scan` determines are part of the scan chain. If scan chain tracing fails, use this parameter to see where the scan chain tracing stops.

Examples

The following set of commands traces all scan chains defined by `create_scan_chain` command:

```
create_scan_chain chain0 -start BG_scan_in_1 -stop BG_scan_out_1
```

```
create_scan_chain chain1 -start BG_scan_in_2 -stop BG_scan_out_2
create_scan_chain chain2 -start BG_scan_in_3 -stop BG_scan_out_3
trace_scan
```

Related Topics

- create_scan_chain
- reorder_scan
- set_lockup_element
- Placing the Design chapter in the *Innovus Stylus Common UI User Guide*
    - Optimizing and Reordering Scan Chains

update_inbound_cell_site

```
update_inbound_cell_site
[-help]
[-n_filler site_name]
[-nn_filler site_name]
[-p_filler site_name]
[-pp_filler site_name]
[-new site_name]
[-original site_name]
[-vertical]
```

Change sites for the cells whose site definitions do not match the site definitions used by Innovus. This is applicable only for specific nodes.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `update_inbound_cell_site` parameter. For a detailed description of the command and all of its parameters, use the man command: `man update_inbound_cell_site`
`-n_filler` *`site_name`*	Specifies the site name of the N-filler cell. *Data_type*: string, optional
`-nn_filler` *`site_name`*	Specifies site name of NN-filler cell. *Data_type*: string, optional
`-p_filler` *`site_name`*	Specifies the site name of the P-filler cell. *Data_type*: string, optional
`-pp_filler` *`site_name`*	Specifies site name of PP-filler cell. *Data_type*: string, optional
`-new` *`site_name`*	Specifies the new site name of the IB cell *Data_type*: string, optional
`-original` *`site_name`*	Specifies the original site name of the IB cell. *Data_type*: string, optional
`-vertical`	Specifies vertical IB transform. *Data_type*: bool, optional

update_pipeline_net_group

```
update_pipeline_net_group
net_group_name
[{-inst instNameListOrRegularExp |
      -start_term termList -end_term termList |
      -net listOfNetnamesOrRegularExp }]
[-start_reserve distInMicrons] [-end_reserve distInMicrons]
```

Modifies the existing net group attributes or converts an existing net group added during floorplanning to be of type pipeline. This command can be used if there are existing pipeline net groups in the design.

Parameters

`-end_reserve` *`ReserveDistanceInMicrons`*	Specifies a distance value from the end points on block boundary to the internal connection inside the block that needs to be added to the last stage distance, to achieve equidistant spacing. The unit is in micron. *Data_type*: float, optional
`-inst` *`instNameListOrRegularExp`*	Modifies net group to add all nets connected to the given instances. The instances must all be flip-flops or latches. All the nets attached to the D and Q pins are put into the PFF net group. *Data_type*: string, optional
`-net` *`listOfNetNamesOrRegularExp`*	Creates net group of all given nets. *Data_type*: string, optional
`-start_reserve` *`ReserveDistanceInMicrons`*	Specifies a distance budget from internal connections inside block to the start point of the PFF net. This value is added to the first stage distance when calculating equidistant spacing for each stage. The unit is in micron. *Data_type*: float, optional
`-start_term` *`termList`*	Modifies net group to add all data nets between the given end points. The PFF-chain is traced from the startTerm, through D and Q terminals of PFF(s) until an endTerm is matched. *Data_type*: string, optional
`-end_term` *`termList`*	Modifies net group to add all data nets between the given end points. The PFF-chain is traced from the startTerm, through D and Q terminals of PFF(s) until an endTerm is matched. *Data_type*: string, optional
`net_group_name`	Specifies the name of net group. *Data_type*: string, required

Example

The following command modifies the existing net group named `netGroup1` between `blockA/DATA_OUT*` and `blockB/DATA_IN*` with a budget distance 10 microns from internal connections inside the blockA to the start point of `DATA_OUT*` :

```
update_pipeline_net_group netGroup1 -start_term blockA/DATA_OUT* -end_term blockB/DATA_IN*
-start_reserve 10
```

Related Information

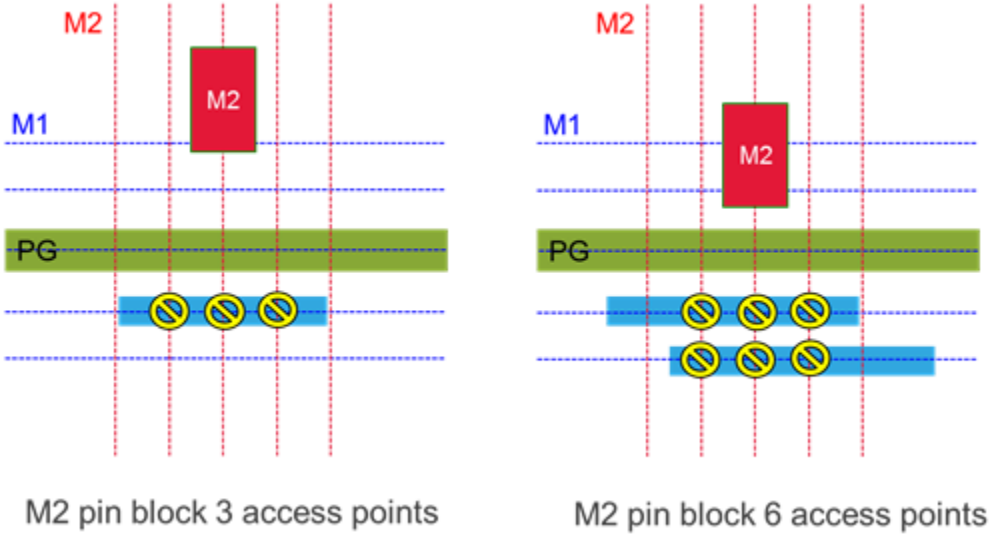
- `create_pipeline_net_group`
- `delete_pipeline_net_group`
- `report_pipelines`

update_vertical_cell_edge_constraint

```
update_vertical_cell_edge_constraint
[-help]
{-file tclFile | -update_db }
[[-check_m2_only ] | [-check_m2_to_m1_only ]]
```

Generates the bottom and top cell edge constraints for 10nm libraries to avoid vertical cell abutment that may cause pin access issue on M1 and M2. The vertical M2 fat PIN or OBS has impact on the abutted cell when it is near to the top and bottom boundaries. It can cause DRC violations between M2 PIN/OBS, block some of the M1 pin access points in the abutted cell, and make router hard to access those M1 PINs.

The following graphic show an example of how fat M2 Pin/OBS may block 3 or 6 access points.



The `update_vertical_cell_edge_constraint` command generates vertical cell edge constraints based on the cell libraries and LEF rules. Placer is aware of these constraints and avoids the bad placement patterns.

Parameters

`-check_m2_only`	Checks M2 to M2 pins only in abutted 2 cells when generating cell edge constraints. *Data_type*: bool, optional
`-check_m2_to_m1_only`	Checks M2 to M1 pins only in abutted 2 cells when generating cell edge constraints. *Data_type*: bool, optional
`-help`	Outputs a brief description that includes type and default information for each `update_vertical_cell_edge_constraint` parameter. For a detailed description of the command and all of its parameters, use the man command: `man update_vertical_cell_edge_constraint`.
`-file` *`tclFile`*	Specifies file name to output vertical tcl constraint. *Data_type*: string, optional
`-update_db`	Updates the constraints to DB. *Data_type*: bool, optional

Note:

- You must specify `-file` or `-update_db`.
- The `-file` option is for the debug mode, and you need to source the tcl file to let placer know about the constraints.
- The `-updated` option is for the standard flow usage. It reads the constraints automatically.

pack_align_macros

```
pack_align_macros
[-help]
```

The `pack_align_macros` command executes the following two steps:

- Aligns macros of the same size (initially placed together) on the entire design.
- Pushes macros (or pack of macros) to core corners and marks them fixed.

The `pack_align_macros` command honors the min space/forbidden space.

Figure 1: Macros before running `pack_align_macros`

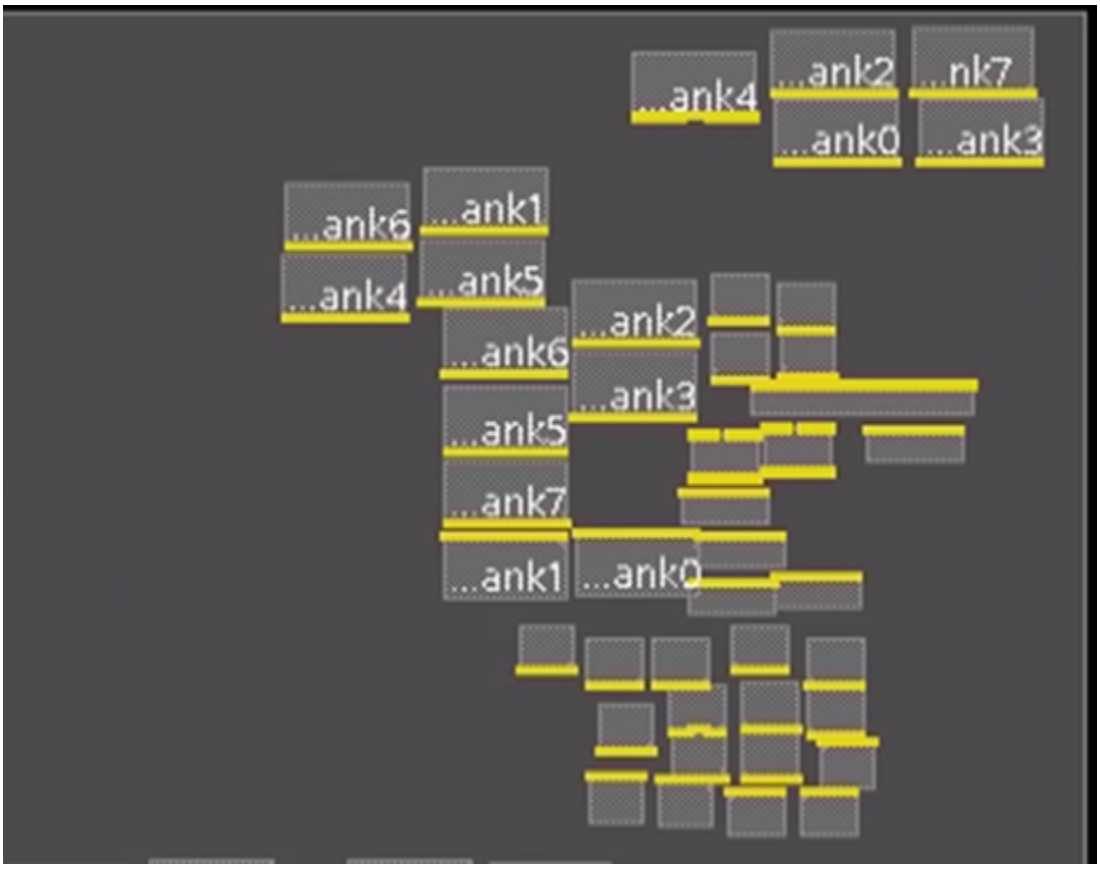
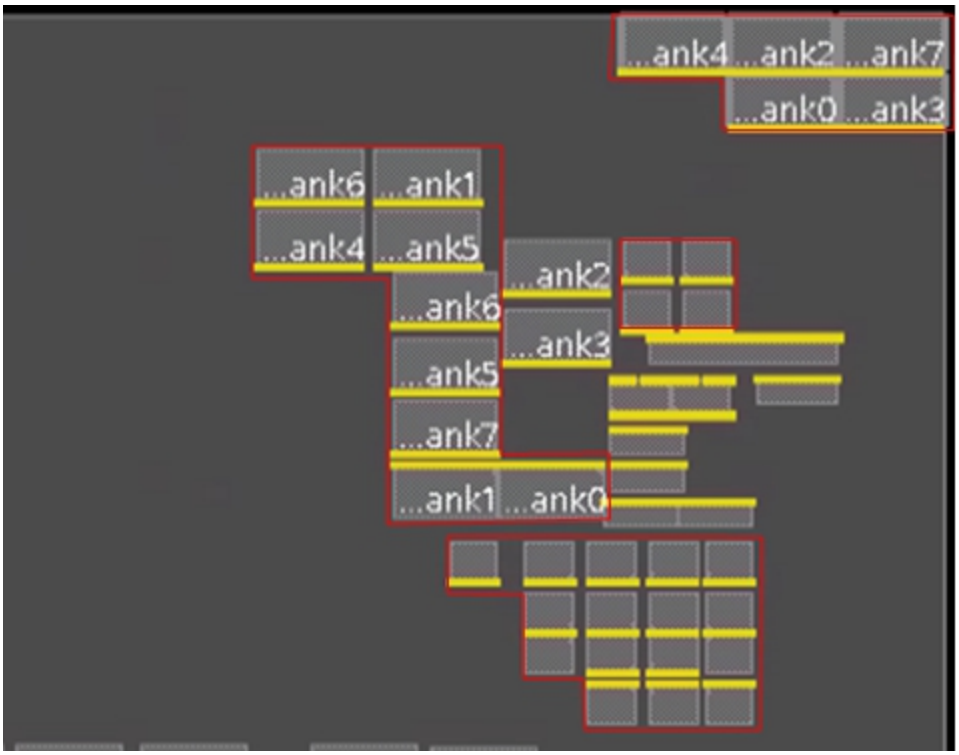
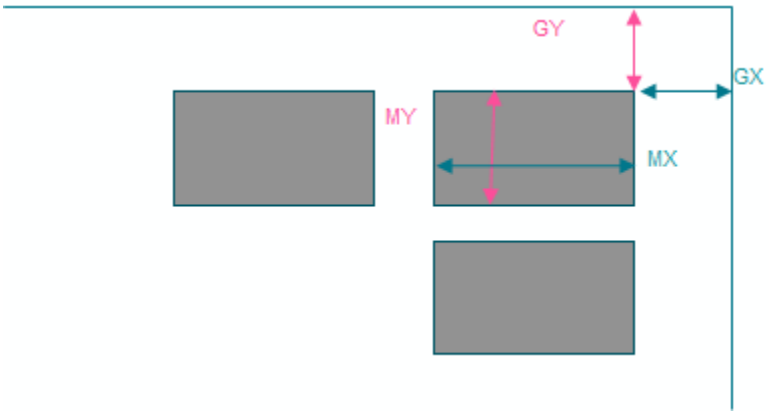


Figure 2: Macros aligned after running `pack_align_macros`



The Figure 1 and Figure 2 show a design before and after running the `pack_align_macros`, respectively. The Figure 2 shows that the push till the core corner takes place when the gap between the core corner and the corner of the nearest macro satisfies the following condition:
GY <= MY And GX <= MX



Parameters

None

26

Power Calculation Commands and Attributes

- encrypt_thermal_file
- get_dynamic_power_simulation
- get_default_switching_activity
- power Category Attributes
- propagate_activity
- query_power_data
- read_activity_file
- read_activity_map_file
- read_power_db
- report_inst_power
- report_power
- report_unannotated_nets
- reset_power_activity
- set_default_switching_activity
- set_dynamic_power_simulation
- set_instance_temperature_file
- set_power
- set_power_calculation_temperature
- set_power_include_file
- set_power_output_dir
- set_switching_activity
- set_twf_attribute
- set_virtual_clock_network_parameters
- write_power_constraints
- write_tcf

encrypt_thermal_file

```
encrypt_thermal_file
[-input ASCII.txt]
[-output $encrypted_conductivity.txt]
```

Specifies to encrypt the file containing the thermal conductivity value for layers, and to protect foundry's thermal conductivity parameters used for power map generation. This encrypted file is then specified with the `report_power -thermal_conductivity_inputs` parameter for generating a tile-based power map file.

Parameters

```
-input ASCII.txt
```

	Specifies the name of the ASCII thermal file that needs to be encrypted. This file contains the thermal conductivity parameters used for power map generation.
	The `-input` parameter supports both layer type and layer name based thermal conductivity file.
	The following is a snippet of the input thermal conductivity file content (layer type based file):
	`Metal 430`
	`Silicon 250`
	`Dielectric 0.6`
	Data_type: string, optional

`-output` *`$encrypted_conductivity.txt`*	

	Specifies the name of the encrypted output thermal file.
	The following is a snippet of the encrypted thermal conductivity file content:
	`Metal $encryption_value`
	`Silicon $encryption_value`
	`Dielectric $ecncryption_value`
	Data_type: string, optional

Examples

- The following command encrypts the specified input file (`format_1.txt`) to generate the encrypted output file (`format_1_encr.txt`):
  ```
  encrypt_thermal_file -input encrypt/format_1.txt -output encrypt/format_1_encr.txt
  ```
  The encrypted file (`format_1_encr.txt`) is then specified for generating a tile-based power map file, as shown below:
  ```
  report_power \
      -out_file                    static.rpt \
      -thermal_power_map_tileS     {5 5}\
      -thermal_power_map_format    stack \
      -thermal_power_map_file      powermap_format1_encr_in.txt \
      -thermal_conductivity_inputs ./encrypt/format_1_encr.txt
  ```

Related Information

- `report_power`
- "Static Power, IRdrop and EM Analysis" in the *Voltus Stylus Common UI User Guide*

get_dynamic_power_simulation

```
get_dynamic_power_simulation
[-help]
[-activity_pattern]
[-period]
[-resolution]
```

Displays the following information about a `set_dynamic_power_simulation` parameter in the system log file and in the system console:

- Parameter name
- Current value

If you do not specify a parameter, the software displays information for all of the `set_dynamic_power_simulation` parameters.
The `get_dynamic_power_simulation` command prints only the user-specified value for the parameters. If a parameter has not been set using the `set_dynamic_power_simulation` command, the `get_dynamic_power_simulation` command does not output any information in the console.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `get_dynamic_power_simulation` parameter. For a detailed description of the command and all of its parameters, use the man command: `man set_dynamic_power_simulation`.
`parameter_names`	Displays information for the specified parameters. You can specify one or more parameters. See `set_dynamic_power_simulation` for description of the parameters you can specify.

Examples

The following command displays the current setting of the `-period` parameter:

```
get_dynamic_power_simulation -period
```

The software displays the following information:

```
-period {20}
```

Related Information

- `power Category Attributes`
- `set_default_switching_activity`
- `set_dynamic_power_simulation`
- "Dynamic Power and IRDrop Analysis" in the *Voltus Stylus Common UI User Guide*

get_default_switching_activity

```
get_default_switching_activity
[-help]
[-clip_activity_to_domain_freq]
[-clock_gates_enable]
[-clock_gates_output]
[-clock_gates_output_ratio]
[-comb_clockgate_ratio]
[-duty]
[-global_activity]
[-icg_ratio]
[-input_activity]
[-macro_activity]
[-period]
[-seq_activity]
```

Displays the following information about a `set_default_switching_activity` parameter in the system log file and in the system console:

- Parameter name
- Current value

If you do not specify a parameter, the software displays information for all of the `set_default_switching_activity` parameters.
The `get_default_switching_activity` command prints only the user-specified value for the parameters. If a parameter has not been set using the `set_default_switching_activity` command, the `get_default_switching_activity` command does not output any information in the console.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `get_default_switching_activity` parameter. For a detailed description of the command and all of its parameters, use the man command: `man set_default_switching_activity`.
`parameter_names`	Displays information for the specified parameters. You can specify one or more parameters. See `set_default_switching_activity` for description of the parameters you can specify.

Examples

The following command displays the current setting of the `-period` parameter:

```
get_default_switching_activity -period
```

The software displays the following information:

```
-period {20}
```

power Category Attributes

The root attributes in the `power` category are:

- power_add_simulation
- power_adjust_input_activity_in_iterations
- power_adjust_macro_activity_in_iterations
- power_analysis_temperature
- power_annotation_detail_report
- power_auto_twf_delay_annotation
- power_average_rise_fall_cap
- power_block_independent_peakpower
- power_bulk_pins
- power_capacity
- power_case_insensitive_mapping
- power_clock_source_as_clock
- power_comprehensive_automapping
- power_constant_override
- power_corner
- power_create_driver_db
- power_current_generation_method
- power_db_name
- power_decap_cell_list
- power_default_frequency
- power_default_slew
- power_default_supply_voltage
- power_disable_clock_gate_clipping
- power_disable_leakage_scaling
- power_disable_static
- power_distribute_switching_power
- power_distributed_combine_report_format
- power_distributed_setup
- power_domain_based_clipping
- power_dynamic_glitch_filter
- power_enable_auto_queue
- power_enable_auto_mapping
- power_enable_disk_mapping
- power_enable_duty_propagation_with_global_activity
- power_enable_dynamic_current_slew_load_interpolation
- power_dynamic_scale_clock_by_frequency
- power_dynamic_scale_clock_by_name
- power_dynamic_vectorless_ranking_methods
- power_enable_dynamic_scaling
- power_enable_flop_state_propagation
- power_enable_generated_clock

- power_enable_input_net_power
- power_enable_interactive_reports
- power_enable_mt_state propagation
- power_enable_multithread_reports
- power_enable_power_target_flow
- power_enable_pba_for_tempus_pi
- power_enable_rtl_dynamic_vector_based
- power_enable_scan_report
- power_enable_slew_based_ccs_pin_cap
- power_enable_state_propagation
- power_enable_tempus_pi
- power_enable_xp
- power_enhanced_blackbox_avg
- power_enhanced_blackbox_max
- power_equivalent_annotation
- power_event_based_leakage_power
- power_external_load_config_file
- power_extraction_tech_file
- power_extractor_include
- power_fanout_limit
- power_force_library_merging
- power_from_x_transition_factor
- power_from_z_transition_factor
- power_generate_activity_mapping_report
- power_generate_current_for_rail
- power_generate_flop_ranking_data
- power_generate_static_report_from_state_propagation
- power_generate_leakage_power_map_based_on_calculated_leakage
- power_grid_libraries
- power_handle_glitch
- power_handle_tristate
- power_hier_delimiter
- power_honor_combinational_logic_on_clock_net
- power_honor_negative_energy
- power_honor_net_activity
- power_honor_non_mission_leakage
- power_hybrid_analysis
- power_ignore_control_signals
- power_ignore_data_phase_for_clock
- power_ignore_end_toggles_in_profile
- power_ignore_glitches_at_same_time_stamp
- power_ignore_inout_pin_cap
- power_include_file

- power_include_initial_x_transitions
- power_include_sequential_clock_pin_power
- power_include_timing_in_current_file
- power_ir_derated_timing_view
- power_keep_clock_gate_ratio_in_iterations
- power_leakage_scale_factor_for_temperature
- power_lib_files
- power_library_preference
- power_match_state_for_logic_x
- power_multibit_flop_toggle_behavior
- power_multi_scenario_simulation
- power_merge_switched_net_currents
- power_method
- power_min_leaf_count
- power_off_pg_nets
- power_output_current_data_prefix
- power_output_dir
- power_partition_count
- power_partition_twf
- power_pin_based_twf
- power_precision
- power_pre_simulation_empty_period
- power_pre_simulation_period
- power_pre_simulation_power_exclude_period
- power_quit_on_activity_coverage_threshold
- power_read_rcdb
- power_relax_arc_match
- power_report_black_boxes
- power_report_idle_instances
- power_report_instance_switching_info
- power_report_instance_switching_list
- power_report_library_usage
- power_report_missing_bulk_connectivity
- power_report_missing_input
- power_report_missing_nets
- power_report_scan_chain_statistics
- power_report_statistics
- power_report_time_display_fraction_digits
- power_report_twf_attributes
- power_reuse_flop_ranking_data
- power_reuse_flop_ranking_data_hier
- power_scale_to_sdc_clock_frequency
- power_scan_chain_activity

- power_scan_chain_name_pattern
- power_scan_control_file
- power_scan_multi_bit_flop_chain_type
- power_settling_buffer
- power_smart_window
- power_split_bus_power
- power_start_time_alignment
- power_state_dependent_leakage
- power_stateprop_ignore_unannot_pins
- power_static_multi_mode_scenario_file
- power_static_netlist
- power_switching_power_on_rise_only
- power_thermal_input_file
- power_thermal_leakage_temperature_scale_table_file
- power_to_x_transition_factor
- power_to_z_transition_factor
- power_transition_factor_based_duty
- power_transition_time_method
- power_twf_delay_annotation
- power_twf_load_cap
- power_use_cell_leakage_power_density
- power_use_fastest_clock_for_dynamic_scheduling
- power_use_lef_for_missing_cells
- power_use_physical_partition
- power_worst_case_vector_activity
- power_worst_step_size
- power_worst_window_count
- power_worst_window_coverage
- power_worst_window_reports
- power_worst_window_size
- power_worst_window_type
- power_use_zero_delay_vector_file
- power_vector_based_multithread
- power_vector_profile_mode
- power_write_bbox
- power_write_db
- power_write_default_pti_files
- power_write_dynamic_currents
- power_write_gui_db
- power_write_profiling_db
- power_write_simulation_db
- power_write_static_currents
- power_x_count_transition_using_3_states

- power_x_transition_factor
- power_z_transition_factor
- power_zero_delay_vector_toggle_shift

To set the value of any root attribute, use the `set_db` command:

```
set_db attribute_name value
```

To get the current value of any root attribute, use the `get_db` command:

```
get_db attribute_name
```

To get the current value of all attributes in the `power` category, use the following command:

```
get_db -category power
```

power_add_simulation

```
power_add_simulation {true | false}
```

Default: `false`

Data_type: bool

Specifies to add simulation if there are no simulation vectors or if all primary sources are not annotated by vectors. This parameter can be used to ensure complete design coverage. Event simulation is done for primary sources like primary input, flops, and macros.

This attribute is only applicable to the event-based power analysis flow (`power_method event_based`).

Related Commands

- `set_power`

power_adjust_input_activity_in_iterations

```
power_adjust_input_activity_in_iterations {true | false}
```

Default: `true`

Data_type: bool

Specifies whether to keep the input activity fixed or automatically adjust the activity value for each iteration of state propagation. This attribute is only applicable to the power target flow (`power_enable_power_target_flow true`).

Related Commands

- `set_power`

power_adjust_macro_activity_in_iterations

```
power_adjust_macro_activity_in_iterations {true | false}
```

Default: `true`

Data_type: bool

Specifies whether to keep the macro activity fixed or automatically adjust the activity value for each iteration of state propagation. This attribute is only applicable to the power target flow (`power_enable_power_target_flow true`).

Related Commands

- `set_power`

power_analysis_temperature

```
power_analysis_temperature temperature
```

Default: " "

Data_type: double

Specifies the temperature for static power calculation.

Note: Static power calculation gets the temperature value of each instance from the following sources (in order of precedence):

- Instance temperature file
- `power_analysis_temperature` attribute
- The current operating condition temperature

Related Commands

- `report_power`

power_annotation_detail_report

```
power_annotation_detail_report {true | false}
```

Default: `false`

Data_type: bool

Specifies to generate a detailed annotation report per instance in the Vector Profiling flow (`power_method vector_profile`). When specified, it creates the `AnnotationDetail.rpt` file in the user-specified output directory.

Related Commands

- `report_power`

power_auto_twf_delay_annotation

```
power_auto_twf_delay_annotation {true | false}
```

Default: `false`

Data_type: bool

Specifies to automatically detect the zero-delay vectors and perform delay annotation. As a result, you do not need to specify additional parameters (`set_db power_use_zero_delay_vector_file` and `read_activity_file -zero_delay`) to indicate the vectors that have delay annotation. This attribute is applicable only to the event-based power analysis flow.

Related Commands

- `report_power`

power_average_rise_fall_cap

```
power_average_rise_fall_cap {true | false}
```

Default: `false`

Data_type: bool

Specifies to use the average of rise and fall capacitance from the liberty file.

Related Commands

- `report_power`

power_block_independent_peakpower

```
power_block_independent_peakpower {true | false}
```

Default: `false`

Data_type: bool

Specifies to identify the worst power interval of each block, and run power analysis of the full design using the worst interval of each block. This attribute allows you to perform power and current analysis for design blocks that have independent peak power intervals.

When set to `false`, power analysis is performed using the common worst interval found using simulation of all vectors together.

Related Commands

- `report_power`

power_bulk_pins

```
power_bulk_pins {bulk_pin_list}
```

Default: `""`

Data_type: string

Defines power and ground bulk LEF pins for the design.

Use this attribute when library cells have bulk pins defined in LEF but the liberty file does not contain power associated with these pins. If a liberty file does not have body bias definition, power analysis does not distribute power to the body bias domain.

Related Commands

- `report_power`

power_capacity

```
power_capacity {low|medium|high}
```

Default: `low`

Data_type: enum

Enables segmented power processing. When this attribute is specified, the software performs concurrent processing of vector simulation annotation, power calculation, and report construction.

This attribute enables performance optimization with some trade-off on memory consumption and runtime. The arguments of `power_capacity` are:

- `low` - reduces memory for a flat run.
- `medium` - further reduces memory with moderate increase in runtime (1.2X).
- `high` - maximum memory reduction at about 2X runtime.

The `power_capacity` attribute is used for reducing memory when processing long simulation vectors. In case of the FSDB and SHM format activity file, the runtime can increase up to 2X.

Related Commands

- `report_power`

power_case_insensitive_mapping

```
power_case_insensitive_mapping {true | false}
```

Default: `false`

Data_type: bool

Specifies whether instance name mapping between the RTL netlist and GATE level netlist in the mapping file is case-sensitive or case-insensitive. When set to `true`, name mappings are case-insensitive.

Related Commands

- `report_power`

power_clock_source_as_clock

```
power_clock_source_as_clock {true | false}
```

Default: `false`

Data_type: bool

Specifies to use the correct clock frequency for activity calculation. If you stop the arrival of the clock signal by setting `set_case_analysis` to `false` , the transition density at the output net of the flop is calculated using the correct clock frequency instead of the default frequency.

Related Commands

- `report_power`

power_comprehensive_automapping

```
power_comprehensive_automapping {true | false}
```

Default: `false`

Data_type: bool

Specifies to perform all-inclusive instance name mapping between the RTL netlist and GATE level netlist. When set to `true`, the software honors all possible naming rules (includes handling of characters like '[' , ']', '.', '/', '\\', '_', ':' and handling of strings like "MBFF", "_reg", "_MB_" etc.) for better annotation coverage.

This feature is applicable to both file-based mapping (`read_activity_map_file -rtl_to_gate | -two_column_mapping_file`) and automatic mapping (`power_enable_auto_mapping`). When `power_comprehensive_automapping` is specified with either `-rtl_to_gate` or `-two_column_mapping_file`, the RTL mapping entries are given priority over the automatic rule-based mapping entries and are used to annotate the design objects.

Related Commands

- `report_power`

power_constant_override

```
power_constant_override {true | false}
```

Default: `false`

Data_type: bool

Specifies to give higher precedence to propagated constants defined using the `set_case_analysis` command. It allows propagated constants to override global activity.

Related Commands

- `report_power`

power_corner

```
power_corner {min|max}
```

Default: `max`

Data_type: string

Defines the library corner (for non-MMMC setup).

- `min` corner means power calculator will use min timing libraries for the power calculation.
- `max` corner means power calculator will use max timing libraries for the power calculation.

Related Commands

- `report_power`

power_create_driver_db

```
power_create_driver_db {true | false}
```

Default: `false`

Data_type: bool

Specifies to determine all the drivers/receivers on the ground net.

Related Commands

- report_power

power_current_generation_method

```
power_current_generation_method {avg | peak}
```

Default: `avg`

Data_type: string

Specifies the current waveform generation method during dynamic power analysis. The two possible arguments of this attribute are:

- `avg` - Specifies to use the average current value of each time step. The output current filename is *`filename`*`.ptiavg`. The default value is `avg`.
- `peak` - Specifies to use the peak current value of each time step. The output current filename is *`filename`*`.ptipeak`. It is recommended to use the `peak` method when using the Composite Current Source Power (CCSP) Liberty files.

Related Commands

- report_power

power_db_name

```
power_db_name filename
```

Default: `""`

Data_type: string

Specifies the name of the binary power database file (`filename.db`).

You can specify this attribute in scenarios where more than one db is generated and you need to track which one was generated in which mode when the restoring the db, or if your design has several blocks and analysis is done on one block all the time and all the databases are named `power.db`.

Related Commands

- report_power

power_decap_cell_list

```
power_decap_cell_list cell_list
```

Default: `""`

Data_type: string

Specifies the physical only cells, such as decap cells, to include in the power report.

Note: This is only supported for Static Power Analysis.

Related Commands

- report_power

power_default_frequency

```
power_default_frequency value
```

Default:

Data_type: double

Specifies to set the default frequency in the MHz unit for net not annotated by TWF. You can specify a numerical value without unit.

Related Commands

- report_power

power_default_slew

```
power_default_slew value
```

Default:

Data_type: string

Specifies to set the default slew in the ps unit for net not annotated by TWF. You can specify a numerical value without unit.

Related Commands

- report_power

power_default_supply_voltage

```
power_default_supply_voltage value
```

Default:

Data_type: string

Specifies to set a default voltage for power nets in a scenario where the power engine cannot determine the voltage of these nets. The default value of this attribute is `1.0`.

Related Commands

- report_power

power_disable_clock_gate_clipping

```
power_disable_clock_gate_clipping {true | false}
```

Default: `true`

Data_type: bool

Specifies to enable clock gate output clipping so that the output transition density of an ICG cell does not exceed the input clock pin transition density.

Related Commands

- report_power

power_disable_leakage_scaling

```
power_disable_leakage_scaling {true | false}
```

Default: `false`

Data_type: bool

Specifies to exclude leakage power during scaling factor computation when target power is specified using the `set_power` command. When this option is set to `true`, the leakage power does not get scaled, and only the internal and switching components are scaled to meet the user-specified total power.

This attribute is relevant only for the `set_power` commands for which total power is specified, and has no effect for the `set_power` command specified with the `-internal`, `-switching`, or `-leakage` parameters.

Related Commands

- `report_power`

power_disable_static

```
power_disable_static {true | false}
```

Default: `true`

Data_type: bool

When set to `true` in the dynamic vector-based or dynamic vectorless flows, the command performs only dynamic analysis and turns-off static power calculation.

Related Commands

- `report_power`

power_distribute_switching_power

```
power_distribute_switching_power {true | false}
```

Default: `false`

Data_type: bool

Specifies to equally distribute the switching power among all drivers if a net has more than one driver. When set to `false`, stores all switching power on one of the net drivers.

Related Commands

- `report_power`

power_distributed_combine_report_format

```
power_distributed_combine_report_format {none | detailed | reduced}
```

Default: `reduced`

Data_type: string

Specifies to generate consolidated XP power reports. This parameter allows you to combine power reports that are stored across multiple partition directories to give a single unified view of power data. The possible arguments of this parameter are:

- `none` - generates distributed reports.
- `reduced` - generates consolidated reports for the following:
  - power.missingdata.* (static and dynamic flows)
  - power.stateprop.* (dynamic flow)
  - power.stats.setpower (dynamic flow)
- `detailed` - generates consolidated reports for the following:
  - power.rpt (static and dynamic flows)
  - power.instpwr.rpt (static flow)
  - power.lib.rpt (static flow)
  - power.pgpinpwr.rpt (static flow)
  - power.pgnet.detailed.rpt
  - power.missingdata.* (static and dynamic flows)
  - power.pgnet.detailed.rpt (static and dynamic flows)

- power.stateprop.* (dynamic flow)
- power.stats.setpower (dynamic flow)
- pg_trigger.txt (dynamic flow)
- trigger.txt (dynamic flow)
- *.totalcurrent (dynamic flow)
- voltus_power.cellstat (static and dynamic flows)
- voltus_power.ptifiles (static and dynamic flows)

Related Commands

- report_power

power_distributed_setup

```
power_distributed_setup file
```

Default: ""

Data_type: string

Specifies a file containing the customized setup details for running power analysis in the distributed mode.

The specified file has 4 columns:

```
INST/CELL <name> hier/block <power.inc>
```

where,

- `INST/CELL`: Keyword indicating cell or instance to be run in the distributed mode. Cell can be useful for repeated DEF blocks.
- *name* : Name of the hierarchical block or instance.
- `hier/block`: Keyword indicating hierarchy or block. `hier` implies that the full hierarchy will be preserved by including the DEFs from top to the block, and `block` implies that only the lower level DEF will be included
- *power.inc* : name of the power include file. This column is optional.

The unspecified instance/cell will be run with the top level as a single group. Each distributed block/instance should have corresponding TWF and SPEF to improve runtime.

Note: Each hierarchy or block will be run independently from other hierarchies/blocks, thus the external load or cell seen at the pin of the hierarchy/block can be different from flat analysis. If the partition boundary analysis is important, distributed processing is not recommended.

Related Commands

- report_power

power_domain_based_clipping

```
power_domain_based_clipping {true | false}
```

Default: `false`

Data_type: bool

Determines the frequency to be used to clip propagated activity that is too high. When set to `false`, the frequency of the fastest clock in the design is used. If you set this attribute to `true`, the fastest clock domain on the net is used to clip propagated activity.

Related Commands

- report_power

power_dynamic_glitch_filter

```
power_dynamic_glitch_filter value
```

Default: `-1`

Data_type: double

Sets the absolute threshold time (in ns) for filtering glitches in net toggles. When you specify a time value for this parameter, glitches in net toggles that are equal to or smaller in width than the specified time value are removed.

Related Commands

- `report_power`

power_enable_auto_queue

```
power_enable_auto_queue {true | false}
```

Default: `false`

Data_type: bool

Specifies to automatically determine the queue for power jobs. This attribute is used in the Voltus-XP distributed power flow with hierarchical DEFs. In the XP flow, partitions are divided into multiple queues depending on the number of CPUs (For example, Default queue: 1 CPU, Small Queue: 2 CPUs, Medium; 4 CPUs, Large : 8 CPUs, and so on.). By default, all power partitions run on the same queue. When the attribute is set to `true`, the blocks can run on different queues depending on the size (number of instances) of the blocks. The `power_enable_auto_queue` attribute will be beneficial when there are uneven power block partitions. For example, if there are some blocks with 20-30M instances and there are some with 3-5M instances. Using this attribute, bigger blocks will be automatically assigned to a bigger queue, resulting in better runtime.

Related Commands

- `report_power`

power_enable_auto_mapping

```
power_enable_auto_mapping {true | false}
```

Default: `false`

Data_type: bool

Specifies to automatically perform instance name mapping between the RTL netlist and GATE level netlist. This attribute allows you to map the RTL vector to the Gate level netlist without a mapping file.

You can the `read_activity_file -name_mapping_rule <filename>` command parameter to specify a file containing customized or design-specific rules.

Related Commands

- `report_power`

power_enable_disk_mapping

```
power_enable_disk_mapping {true | false}
```

Default: `false`

Data_type: bool

Specifies to enable disk caching of the toggles. Therefore, the toggles are stored in the disk instead of keeping them in memory. This has made large vectors or long simulation duration runs possible even with lesser available memory. This attribute is only applicable to the event-based power analysis flow (`power_method event_based`) in the XP mode.

Example:

The following command specifies to enable disk caching of toggles:

```
set_db power_method event_based power_enable_disk_mapping true power_enable_xp true
```

Related Commands

- `report_power`

power_enable_duty_propagation_with_global_activity

```
power_enable_duty_propagation_with_global_activity {true | false}
```

Default: `false`

Data_type: bool

Specifies to propagate duty cycle with the global switching activity setting for all data nets in the dynamic vectorless analysis flow.

Related Commands

- report_power

power_enable_dynamic_current_slew_load_interpolation

```
power_enable_dynamic_current_slew_load_interpolation {true | false}
```

Default: `false`

Data_type: bool

Specifies to use the charge-based slew load interpolation. When this attribute is set to `true`, Power Analysis takes dynamic currents from the 4 nearest slew/load values and performs either a 4-way (lower slew, lower load, upper slew, upper load) or a 2-way (lower slew, upper slew OR lower load, upper load) interpolation to calculate the dynamic current waveform. The attribute allows you to derive a more realistic dynamic current waveform based on the exact slew load values from the .lib dynamic current table.

Related Commands

- report_power

power_dynamic_scale_clock_by_frequency

```
power_dynamic_scale_clock_by_frequency { freq_Afactor_Afreq_Bfactor_Bfreq_Cfactor_C... }
```

Default: `""`

Data_type: string

Specifies the scale factors for the clock frequency values. This parameter allows you to perform frequency scaling for clocks running at relatively low frequencies, thereby ensuring higher toggle coverage for limited simulation period. The parameter is applicable only to the dynamic vectorless flow. The following units are supported for frequency: Hz, KHz, MHz, and GHz.

Related Commands

- report_power

power_dynamic_scale_clock_by_name

```
power_dynamic_scale_clock_by_name { clk_A factor_A clk_B factor_B clk_C factor_C ... }
```

Default: `""`

Data_type: string

Specifies the scale factors for the low frequency clocks. This parameter allows you to perform frequency scaling for clocks running at relatively low frequencies, thereby ensuring higher toggle coverage for limited simulation period. The parameter is applicable only to the dynamic vectorless flow. The following units are supported for frequency: Hz, KHz, MHz, and GHz.

Related Commands

- report_power

power_dynamic_vectorless_ranking_methods

```
power_dynamic_vectorless_ranking_methods {load | clock | vector_activity}
```

Default: `""`

Data_type: string

Specifies to change the ranking method from the default to the following:

- `load` - Prioritizes scheduling cells with higher fanout
- `clock -` Prioritizes scheduling cells associated with faster clocks
- `vector_activity` - Prioritizes scheduling cells with higher activity

Related Commands

- `report_power`

power_enable_dynamic_scaling

```
power_enable_dynamic_scaling {true | false}
```

Default: `false`

Data_type: bool

Enables `set_power` scaling for dynamic power analysis. The power and scaling specifications are used to scale dynamic current on both the power and ground currents of the specified instance/cell.

This attribute honors the user-defined average (static) power for leaf-level instances and hierarchies (instances/cells/modules) along with power net specification in the dynamic current scaling flows such that the average of the dynamic current is equal to the assigned average power. When you specify this attribute, the average of the dynamic current of these leaf-level instances will match the scaled static power numbers.

⚠ This attribute is not supported in the state-propagation-based vectorless flow (`power_enable_state_propagation true`). It is applicable only to the probability-based vectorless flow.

Related Commands

- `set_power`
- `report_power`

power_enable_flop_state_propagation

```
power_enable_flop_state_propagation {true | false}
```

Default: `false`

Data_type: int

Enables accurate clock state propagation for sequential cells. When this attribute is set to `true`, flops propagate in a manner that toggles on the output pins are in sync with the clock. This attribute helps to align the output pins' toggles with the clock toggles.

Related Commands

- `set_power`
- `report_power`

power_enable_generated_clock

```
power_enable_generated_clock {true | false}
```

Default: `true`

Data_type: bool

Specifies to get generated clock frequency during activity propagation for static power analysis.

Related Commands

- set_power

power_enable_input_net_power

```
power_enable_input_net_power {true | false}
```

Default: false

Data_type: bool

Specifies to calculate the switching power of input nets.

Related Commands

- set_power

power_enable_interactive_reports

```
power_enable_interactive_reports {true | false}
```

Default: false

Data_type: bool

Specifies to generate multiple dynamic reports based on the power analysis run done for a netlist. When this attribute is set to true, you can generate various types of power reports interactively using the report_power command without the need to perform power calculation on the same netlist again.
The power_enable_interactive_reports attribute is useful when you need to generate reports for the unaltered netlist within the same power analysis session.

This attribute is applicable only to the event-based power analysis flow with area-based partitioning (power_use_physical_partition) in the XP mode.

Example:

The following command specifies to enable interactive generation of power reports:

```
set_db power_method event_based power_use_physical_partition true power_partition_count 2 power_enable_interactive_reports true
power_enable_xp true
```

Related Commands

- set_power

power_enable_mt_state propagation

```
power_enable_mt_state_propagation {true | false}
```

Default: true

Data_type: bool

Specifies to enable multi-threading in state propagation to help improve the runtime for large simulations. For example, when the attribute was specified for a design with 300 million instances and simulation duration of 120 microseconds, the runtime has improved from 21 hours to 1.5 hours.

This attribute is only applicable to the event-based power analysis flow (power_method event_based).

Related Commands

- set_power

power_enable_multithread_reports

```
power_enable_multithread_reports {true | false}
```

Default: false

Data_type: bool

Specifies to enable multi-threading for generation of power calculation reports. When set to true, it improves the memory and runtime for power reporting in

large designs with more than 100 million instances.

Related Commands

- set_power

power_enable_power_target_flow

```
power_enable_power_target_flow {true | false}
```

Default: false

Data_type: bool

Specifies to enable the power target flow. When set to true, the tool allows you to specify power targets for full-chip power analysis. You can use the set_power command to specify the power targets for full-chip, physical block, logical block, and power/ground net. When these targets are specified, Voltus will automatically scale the toggle rates of clock gate ratio, input, macro, and sequential activities to meet the specified power targets.

Related Commands

- set_power

power_enable_pba_for_tempus_pi

```
power_enable_pba_for_tempus_pi {true | false}
```

Default: false

Data_type: bool

Specifies to enable the Path-Based Analysis (PBA) for slack calculation in the Tempus Power Integrity (Tempus PI) flow. By default, the Graph-Based Analysis (GBA) is used. This attribute is only applicable to the state-propagation-based vectorless flow (set_db power_method dynamic_vectorless power_enable_state_propagation true).

Related Commands

- set_power

power_enable_rtl_dynamic_vector_based

```
power_enable_rtl_dynamic_vector_based {true | false}
```

Default: false

Data_type: bool

Specifies to take an RTL or partial VCD/FSDB file as input and use that for dynamic vector-based flow. This attribute allows you to estimate current even for instances which are missing from the activity file.

You must use this attribute in conjunction with the power_method dynamic_vectorbased attribute.

Related Commands

- set_power

power_enable_scan_report

```
power_enable_scan_report {true | false}
```

Default: false

Data_type: bool

Specifies to generate all the scan mode analysis reports. The following reports will be generated:

- voltus_power.scanff.duplicated.rpt - flip-flops that are specified in multiple chains

- `voltus_power.scanff.missing.rpt` - flip-flops that are missing in the scan control file but exist in the design
- `voltus_power.scanff.nonassigned.rpt` - flip-flops that do not exist in the design and are dropped from the scan control file
- `voltus_power.scanff.nonscan`.rpt - assigned flip-flops that have no scan-out pin
- `voltus_power.scanff.nonswitched.rpt` - non-switching flip-flops in the scan control file
- `voltus_power.scanff.rpt` - detailed scan chain report containing global statistics for all scan chains and statistics for each scan chain in the design
- `voltus_power.scanff.stateconflict.rpt` - flip-flops that have conflicting state assignment

Related Commands

- `report_power`

power_enable_slew_based_ccs_pin_cap

```
power_enable_slew_based_ccs_pin_cap {true | false}
```

Default: `false`

Data_type: bool

Specifies to use the Composite Current Source (CCS) capacitance value based on the slew given in the input pin section of the library cell in the Liberty file. If it is set to `true`, the tool honors the CCS pin capacitance; if it is set to `false`, it takes `max_capacitance` from .lib. This attribute can be used when timing windows and/or a Tempus timing session is not available.

Related Commands

- `set_power`

power_enable_state_propagation

```
power_enable_state_propagation {true | false}
```

Default: `false`

Data_type: bool

Specifies to enable the vectorless state-propagation-based dynamic analysis flow.

For more information, refer to the **State-Propagation-Based Vectorless Methodology** section in the " *Dynamic Power and IRDrop Analysis* " chapter of the *Voltus User Guide*.

Related Commands

- `set_power`

power_enable_tempus_pi

```
power_enable_tempus_pi {true | false}
```

Default: `false`

Data_type: bool

Specifies to enable the Tempus PI analysis flow. When this attribute is set to `true`, data path scheduling is timing-aware, that is, scheduling is based on timing slacks. This attribute is only applicable to the state-propagation-based vectorless flow (`set_db power_method dynamic_vectorless power_enable_state_propagation true`).

For more information, refer to the **Tempus Power Integrity Analysis** chapter in the *Voltus User Guide*.

Related Commands

- `set_power`

power_enable_xp

```
power_enable_xp {true | false}
```

Default: `false`

Data_type: bool

Specifies to run standalone report generation in the XP mode.

Related Commands

- set_power

power_enhanced_blackbox_avg

```
power_enhanced_blackbox_avg {true | false}
```

Default: `false`

Data_type: bool

Specifies to use the average toggle rate of related inputs for enhanced blackbox propagation.

Related Commands

- set_power

power_enhanced_blackbox_max

```
power_enhanced_blackbox_max {true | false}
```

Default: `false`

Data_type: bool

Specifies to use the maximum toggle rate of related inputs for enhanced blackbox propagation.

Related Commands

- set_power

power_equivalent_annotation

```
power_equivalent_annotation {true | false}
```

Default: `false`

Data_type: bool

Specifies to enable equivalent annotation for Q or Qbar pins when only one of them is annotated.

Related Commands

- set_power

power_event_based_leakage_power

```
power_event_based_leakage_power {true | false}
```

Default: `false`

Data_type: bool

Specifies to calculate state-dependent leakage power. When this attribute is set to `true`, it calculates leakage power and current based on the state of an instance at each time point.

For example, at time 0ns, instance instA is in state !I ZN (0 on input, 1 on output), and at time 71ns, the instance is in state I !ZN (1 on input, 0 on output). The

leakage power is calculated based on the different states of instA at different time points.

Related Commands

- set_power

power_external_load_config_file

```
power_external_load_config_file filename
```

Default: None

Data_type: string

Specifies to honor the user-specified load capacitance values for power calculation. This attribute allows you to specify an external load capacitance file when you have a timing window file without the EXTERNAL_LOAD constructs.

File Format:

The format of the user-specified input file is:

```
INST|CELL    blockName    external_load_flle
```

Here, the format of *external_load_flle* is:

```
NET|PORT       name       load_value
```

Examples:

Configuration file:

```
INST         TOP          top_wcard.load
```

External load file (top_wcard.load):

```
UNIT pF
NET   result     0.02
NET   overflow   0.02
```

Related Commands

- set_power

power_extraction_tech_file

```
power_extraction_tech_file filename
```

Default: ""

Data_type: string

Specifies the extraction technology file to be used for top-level power-grid extraction.

Related Commands

- set_power

power_extractor_include

```
power_extractor_include filename
```

Default: ""

Data_type: string

Specifies the extractor (ZX) commands and variables in the form of an include file. This parameter is applicable only for the standalone report generation flow in the XP mode (power_enable_xp true).

Related Commands

- set_power

power_fanout_limit

```
power_fanout_limit value
```

Default: `-1`

Data_type: int

Specifies the fanout limit for nets. If an instance has nets more than or equal to the fanout limit, its switching/internal/leakage power is taken to be zero.

Related Commands

- set_power

power_force_library_merging

```
power_force_library_merging {true | false}
```

Default: `false`

Data_type: bool

Specifies to force merge PGVs even if there are some minor conflicts in the merging libraries. This parameter is applicable only for the standalone report generation flow in the XP mode (`power_enable_xp true`).

Related Commands

- set_power

power_from_x_transition_factor

```
power_from_x_transition_factor value
```

Default: `0.5`

Data_type: double

Specifies how transitions from X to 0/1 are counted.

Related Commands

- set_power

power_from_z_transition_factor

```
power_from_z_transition_factor value
```

Default: `0.25`

Data_type: double

Specifies how transitions from Z to 0/1 are counted.

Related Commands

- set_power

power_generate_activity_mapping_report

```
power_generate_activity_mapping_report  {true | false}
```

Default: `false`

Data_type: bool

Specifies to generate the `activity_mapping.rpt` report in the vector-based dynamic power flow. This report gives the list of signal name mappings between the Golden (RTL netlist - input name in the vector file) and Revised (Gate netlist - annotated design object name) netlists that were used in the flow. The report

displays the list of annotated and unannotated objects and helps to debug the mapping file for unannotated design objects.

`activity_mapping.rpt` supports activity mapping done through the following methods:

- User-specified mapping file (`read_activity_map_file -two_column_mapping_file`)
- Conformal-generated RTL mapping file (`read_activity_map_file -rtl_to_gate`)
- Voltus auto-generated mapping file (`power_enable_auto_mapping`).

Related Commands

- `set_power`

power_generate_current_for_rail

```
power_generate_current_for_rail railnames
```

Default: `""`

Data_type: string

Generates current files for the specified rail.

By default, the command generates current files for all power rails.

Related Commands

- `set_power`

power_generate_flop_ranking_data

```
power_generate_flop_ranking_data directoryname
```

Default: `""`

Data_type: string

Specifies the name of the directory where the flop ranking data is saved.

Related Commands

- `set_power`

power_generate_static_report_from_state_propagation

```
power_generate_static_report_from_state_propagation  {true | false}
```

Default: `false`

Data_type: bool

Specifies to directly generate the static power analysis result based on the RTL-VCD input and state-propagation. When this attribute is specified, the RTL vector power analysis flow does not require the generation of a TCF/FSDB file. This attribute is applicable only to the RTL vector-based dynamic power analysis flow.

Related Commands

- `set_power`

power_generate_leakage_power_map_based_on_calculated_leakage

```
power_generate_leakage_power_map_based_on_calculated_leakage  {true | false}
```

Default: `false`

Data_type: bool

Specifies to scale the instance leakage power by directly multiplying the user-specified scaling factor with the final calculated leakage power value, instead of scaling the leakage power value from the input dotlib.

If this parameter is not specified, the user-specified scaling factor will be applied on the leakage power values from the input dotlib.

Related Commands

- set_power

power_grid_libraries

```
power_grid_libraries library_list
```

Default: ""

Data_type: string

Specifies the name of the Cadence power cell libraries (.cl). You can also specify a list of defined power-grid libraries having decap cells tagged in them.

Note: When specifying a list of PGVs, you must specify the technology library as the first PGV library in the list, followed by the standard and macro PGV libraries. The selection between the standard and macro PGV library is order based. Therefore, if multiple power-grid libraries are specified for a cell, the power engine maps the cell with the first library after the technology library.

Related Commands

- set_power

power_handle_glitch

```
power_handle_glitch {true | false}
```

Default: false

Data_type: bool

A transition is defined as a glitch when the time difference between two sequential toggles are less than half of the rise transition time + fall transition time in the vector-based static power calculation. Both static and dynamic current are derated for glitch transitions. The same derate factor is used for both power and current.

If set to true, glitch power will be identified and calculated according to this definition and it will be reported in a separate column. Switching or internal power calculation does not include power due to glitches.

If set to false, glitch power will not be identified, and the respective power will be calculated as normal signal switches. Hence, the total power will be a little higher.

Related Commands

- set_power

power_handle_tristate

```
power_handle_tristate {true | false}
```

Default: false

Data_type: bool

When set to true, all tristate device enable pin values will be taken into account when determining the propagation of the activity through tristate gates.

Related Commands

- set_power

power_hier_delimiter

```
power_hier_delimiter character
```

Default: None

Data_type: string

Specifies the hierarchical delimiter in the DEF file. You can use this attribute to override the existing hierarchical delimiter.

Related Commands

- set_power

power_honor_combinational_logic_on_clock_net

```
power_honor_combinational_logic_on_clock_net {true | false}
```

Default: `true`

Data_type: bool

Specifies the behavior of combination logic on clock net.

- `true` (default) - specifies to honor activity propagation through combinational logic.
- `false` - specifies that the combination logic will be treated like a combinational clockgate.

Related Commands

- set_power

power_honor_negative_energy

```
power_honor_negative_energy {true | false}
```

Default: `true`

Data_type: bool

When set to `true`, specifies that the software will keep negative internal energy numbers from the `.lib` internal power table. When set to `false`, negative power is treated as 0.

Related Commands

- set_power

power_honor_net_activity

```
power_honor_net_activity {true | false}
```

Default: `true`

Data_type: bool

Specifies to use activity from the `Net` section of the Switching Activity Interchange Format (SAIF) file for power calculation and optimization, and to ignore the pin-based state-dependent path delay (SDPD) activity. The default value of this attribute is `true`.

When this attribute is set to `false`, the power calculation engine can use both the net-based and pin-based activity. In this case, precedence will be given to activity from net or pin that appears later in the SAIF file.

Related Commands

- set_power

power_honor_non_mission_leakage

```
power_honor_non_mission_leakage {true | false}
```

Default: `false`

Data_type: bool

Supports modeling of well-to sub-leakage information when the circuit power supply is off for the level shifter cells. This is referred to as the non-mission mode leakage power calculation. The leakage power for a level shifter cell depends on the state of each PG pin. The Liberty file contains sections for PG pin conditions of each mode/state and non-mission mode leakage power for each PG pin of that mode.

When this attribute is set to `true`, Voltus identifies the pg_pin mode and assigns non-mission mode leakage power associated with the mode.

The use model of the non-mission mode leakage power calculation feature is:

```
power_honor_non_mission_leakage true
power_off_pg_nets {VDD VDDO}
```

Example:
Let us consider a level shifter cell with the PG mode "`retained_VDDO_off`" and the following `pg_pin_condition`:

```
pg_setting_value (retained_VDDO_off) {
pg_pin_condition : "!VBB*VDD*VDDI ";
pg_pin_active_state (VBB, V1);
pg_pin_active_state (VDD, V2);
pg_pin_active_state (VDDI, V2);
```

When you specify a list of PG nets to be turned off along with the `power_honor_non_mission_leakage` attribute, all cells/instances that have the PG state "`retained_VDDO_off`" associated with the specified PG pin condition, will use the corresponding non-mission leakage power.

Related Commands

- `set_power`

power_hybrid_analysis

```
power_hybrid_analysis {true | false}
```

Default: `false`
Data_type: `bool`

Specifies to perform both vector-based and vectorless power analysis in a single session.

Related Commands

- `set_power`

power_ignore_control_signals

```
power_ignore_control_signals {true | false}
```

Default: `true`

Data_type: bool

Specifies that control signals will be ignored when propagating activity. It is recommended to set this parameter to `true`. When set to `true`, the control pins (reset, preset) will not impact the sequential instance's output pin activity during activity propagation.

Related Commands

- `set_power`

power_ignore_data_phase_for_clock

```
power_ignore_data_phase_for_clock {true | false}
```

Default: `false`

Data_type: bool

Specifies to ignore data frequency in the clock path for the clock instances in the design. When this attribute is set to `true`, the software propagates only the fastest clock pin frequency instead of the data pin frequency for activity propagation through clock network. In the default mode, the software picks the fastest frequency between the data phase and clock phase on that pin.

Related Commands

- set_power

power_ignore_end_toggles_in_profile

```
power_ignore_end_toggles_in_profile {true | false}
```

Default: false

Data_type: bool

By default, the software profiles the toggles in the user-selected vector time window by steps including the toggles at the start/end boundary of the window. In each step, the toggles on the start boundary edge of the step is counted in the current step; the toggles on the end boundary edge is counted in the next step.

When this attribute is set to false, the toggles on the end boundary edge is also counted in the last step, as there is no next step to it. This may lead to double count of the power profile for the last step, especially for zero-delay vector profiling.

When this attribute is set to true, the software ignores the toggle on the end boundary of the last step.

Related Commands

- set_power

power_ignore_glitches_at_same_time_stamp

```
power_ignore_glitches_at_same_time_stamp {true | false}
```

Default: true

Data_type: bool

Specifies to ignore glitches at the same transient time in the vector-based static and dynamic power analysis flow. When this attribute is set to true, only the last signal value change is honored, and all the preceding value changes at the same time stamp are ignored. The last signal value will overwrite the previous value for the same signal at the same time stamp, so the last value will represent the state of the signal at that time stamp.

Related Commands

- set_power

power_ignore_inout_pin_cap

```
power_ignore_inout_pin_cap {true | false}
```

Default: false

Data_type: bool

When set to true, ignores the bidirectional pin capacitances (direction : inout) defined for I/O cells in the .lib file, when calculating switching power and internal power; this also includes the default capacitance value for inout pins.

Related Commands

- set_power

power_include_file

```
power_include_file file
```

Default: " "

Data_type: string

Specifies a power analysis include file. You can use this file to specify certain commands of the dynamic engine which do not have a Voltus equivalent. This attribute is used to pass the legacy power analysis commands and report_power command options only in the dynamic mode (vectorbased and vectorless). If the power_method attribute is set to static, the power_include_file attribute is ignored. The power_include_file attribute is required only if the power_static_netlist attribute is set to def.

Related Commands

- set_power

power_include_initial_x_transitions

```
power_include_initial_x_transitions {true | false}
```

Default: true

Data_type: bool

Controls how the initial X state toggle should be counted. The initial X state means the state at time 0.

When set to true, this means that the software counts the power caused by the X->0 or X->1 toggle. When set to false, the software does not count the power caused by the initial X state toggle.

Related Commands

- set_power

power_include_sequential_clock_pin_power

```
power_include_sequential_clock_pin_power {true | false}
```

Default: false

Data_type: bool

Reports the clock pin power of flip-flops as part of the clock network power.

Related Commands

- set_power

power_include_timing_in_current_file

```
power_include_timing_in_current_file {true | false}
```

Default: false

Data_type: bool

Specifies to write timing bits for non-switching instances in the current files during the state-propagation-based flow. When specified, rail analysis will report effective instance voltage (EIV) for both switching and non-switching instances. If the parameter is set to false, rail analysis will report EIV for the switching instances only.

Related Commands

- set_power

power_ir_derated_timing_view

```
power_ir_derated_timing_view view_name
```

Default: " "

Data_type: string

Specifies the name of the timing view used to calculate timing slack and to rank the flops in the Tempus PI flow. This attribute must be specified if there are multiple views.

The power_ir_derated_timing_view attribute defines the timing view used to detect the IR-sensitive paths using a lower voltage than the actual analysis. This lower voltage simulates the IR-drop for each instance in the design. The software uses this view to prioritize the end-points that need to be prioritized for switching.

This attribute is only applicable to the state-propagation-based vectorless flow (set_db power_method dynamic_vectorless power_enable_state_propagation true).

Related Commands

- set_power

power_keep_clock_gate_ratio_in_iterations

```
power_keep_clock_gate_ratio_in_iterations {true | false}
```

Default: false

Data_type: bool

Specifies whether to keep the clock gate output (CGO) fixed or automatically adjust the value for each iteration of state propagation. This attribute is only applicable to the power target flow (power_enable_power_target_flow true).

Related Commands

- set_power

power_leakage_scale_factor_for_temperature

```
power_leakage_scale_factor_for_temperature view_name
```

Default: 1.0

Data_type: double

Performs a linear scaling of the leakage power for temperature in all libraries. This replaces the k-factor for temperature in the .lib file. For process and voltage, the scaling will continue to be based on the corresponding k-factors in the .lib file. A value of 0.8 scales the leakage power in the library to 80 percent. A value of 1.2 scales the leakage power in the library to 120 percent.

Related Commands

- set_power

power_lib_files

```
power_lib_files in_file <file1[file2] ...
```

Default: ""

Data_type: string

Specifies the name of the timing library file(s) for dynamic power analysis. This attribute allows you to run the dynamic power analysis flow without reading a dotlib into design database to reduce background memory consumption. You must have an external TWF file to run this flow as the timing engine will not be invoked from within Voltus. This flow is recommended for large designs where front-end processing of timing libraries is high. Some of the database query commands will not work if this attribute is used.

Related Commands

- set_power

power_library_preference

```
power_library_preference {voltage | ecsm_ccsp}
```

Default: voltage

Data_type: string

Instances of the same cell can be attached to different power domains in a design. If you have specified different PVT libraries, then the software will by default bind each instance to the closest library definition.

You can use this attribute to select either the closest Effective Current Source Model (ECSM)/Composite Current Source-power (CCSP), or the closest voltage library for the instance:

- voltage: when specified, enables library binding to the closest voltage library.

- `ecsm_ccsp`: when specified, enables library binding to the closest ECSM/CCSP library.

Related Commands

- `set_power`

power_match_state_for_logic_x

```
power_match_state_for_logic_x value
```

Default: `0`

Data_type: string

Controls how the logic X will be evaluated in the boolean function of the '`when`' state of a power table.

The attribute uses the following arguments:

- `0` - regards X as 0
- `1` - regards X as 1
- `x/X` - regards X as neither `0` nor `1`, that is, whenever there is any X logic in it, the boolean function will be evaluated `0` (false).

Related Commands

- `set_power`

power_multibit_flop_toggle_behavior

```
power_multibit_flop_toggle_behavior { simultaneous | independent | sbff }
```

Default: `independent`

Data_type: enum

Specifies the toggle behavior of multi-bit flip-flops (MBFF). The possible arguments are:

- `simultaneous`: The multiple bits of the MBFF will toggle or not toggle at the same time. The MBFF will be considered as one flip-flop.
- `independent`: The multiple bits of the MBFF will toggle independently. The MBFF will be considered as one flip-flop.
- `sbff`: The MBFF will be considered as multiple Single-Bit Flip-Flops (SBFFs). Each bit will be considered as one flip-flop, and will toggle independently.

Related Commands

- `set_power`

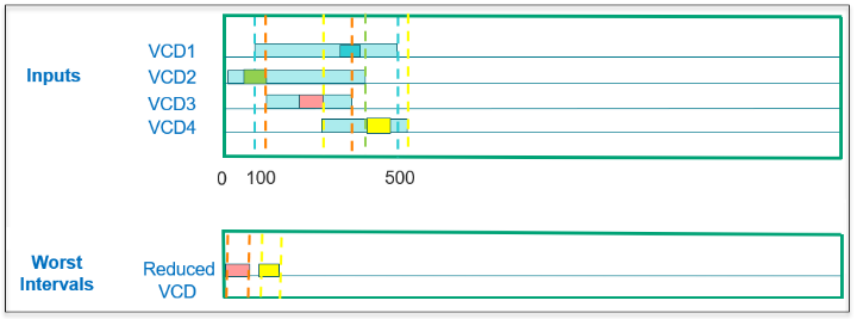
power_multi_scenario_simulation

```
power_multi_scenario_simulation {true | false}
```

Default: `false`

Data_type: bool

Specifies to create a long vector by combining multiple vectors and to identify the worst power intervals across the combined vector. The long vector includes all scenarios that are used sequentially and are non-overlapping.



This attribute allows you to perform power and current analysis using worst peak power interval across multiple simulation vectors.

Related Commands

- `set_power`

power_merge_switched_net_currents

```
power_merge_switched_net_currents {true | false}
```

Default: `false`

Data_type: bool

Specifies to merge switched net currents into always-on currents. For designs with a large number of switched nets, this attribute helps improve performance by merging switched net instance currents with always-on instance currents. However, when this attribute is set to `true`, you will not be able to use the generated current files to run power-up analysis.

Related Commands

- `set_power`

power_method

```
power_method { static | dynamic_vectorless | dynamic_vectorbased | dynamic_mixed_mode | event_based | vector_profile }
```

Default: `static`

Data_type: enum

Specifies the type of analysis to be performed. You can perform two types of power analysis, namely static and dynamic.

- `static` - specifies to perform static power analysis to calculate average power.

Dynamic power analysis can be a vector-based or vectorless approach.

- `dynamic_vectorbased` - uses the full VCD output of a logic simulator to identify the instances that are switching and when they switch.
- `dynamic_vectorless` - uses the timing arrival window information from a static timing analysis tool to determine when instances switch. A timing arrival window describes when a signal can change within a clock cycle. An additional algorithm then determines the instances that switch.
- `dynamic_mixed_mode` - specifies to combine the vector-based and vectorless (mixed) dynamic analysis. This flow can be used in the following cases:
  - **case 1**: vectors are not available for every block of a full-chip design.
    In this case, the software uses the vectorless methodology for blocks with incomplete vectors
  - **case 2**: blocks with vectors having unannotated combinational/ sequential logic.
    In this case, the software performs state propagation for the missing combinational/ sequential logic. It uses the vectorless methodology for a logic cone that does not have the primary inputs or flops annotated from the vector, and then performs state propagation

    To enable the dynamic mixed mode analysis flow, you must specify the following commands:

    ```
    set_default_switching_activity -clock_gates_output 2.0 -global_activity <activity>
    set_db power_method dynamic_mixed_mode
    read_activity_file -scope <> -block <> -> (for available blocks)
    ```
- `event_based -` specifies to enable event-based power analysis. This method supports multiple power analysis flows simultaneously. When specified, the

tool performs accurate state-based power estimation for all events of a scenario in a design. Event-Based Power Analysis is applicable to all event-based vectors, such as VCD, FSDB, SHM, and PHY. You can specify this method to generate and analyze multiple types of power analysis flow reports in a single run.

The following table shows the additional options required for the different flows:

Power Analysis Flows	Additional Options for `-method event_based`
Static	`no additional option required`
Average Power & Current	`-write_static_currents`
Dynamic Vectorbased	`-write_dynamic_currents`
Mixed Mode	`-add_simulation`
Dynamic Vectorless	
Profiling	`-worst_step_size`

- `vector_profile` - specifies to enable vector profiling. This is used to identify windows with maximum activity and power which can then be used to drive dynamic vector-based power analysis.

Related Commands

- `set_power`

power_min_leaf_count

`power_min_leaf_count` *value*

Default: `0`

Data_type: int

Specifies the minimum number of leaf instances required for writing the hierarchy data in the profiling database. It allows you to restrict the number of hierarchies for which waveforms are reported in the profiling database when using both the `power_write_profiling_db` attribute and `report_power -hierarchy` parameters.

Any hierarchy whose leaf instance count in its complete sub-tree is less than the `power_min_leaf_count` value will be skipped from getting reported in the profiling database. This attribute can be used to skip very small hierarchies and as a result save on the database writing time and disk space.

Related Commands

- `set_power`

power_off_pg_nets

`power_off_pg_nets` *net_list*

Default: `""`

Data_type: string

The name of the power rails that will be turned off. The net name can contain instance hierarchy if appropriate. The software will take this data into account when figuring out the list of power gates which are switched off.

Note: A net is turned off if all connected power gates are turned off by control signals. When a net is off, all of the instances connected to the net will only contribute leakage current into the connecting nets.

Related Commands

- `set_power`

power_output_current_data_prefix

```
power_output_current_data_prefix prefix
```

Default: ""

Data_type: string

Specifies a prefix to the static or dynamic current files generated by the software. When power analysis is done in the multi-CPU mode, the output current data directory name and the underlying current files will also use the same prefix by default.

Related Commands

- set_power

power_output_dir

```
power_output_dir directory
```

Default: ""

Data_type: string

Specifies the name of the directory that power information will be output to.

Related Commands

- set_power

power_partition_count

```
power_partition_count value
```

Default: 0

Data_type: integer

Specifies the number of physical partitions or jobs for a distributed run in the area-based partitioning flow. This attribute is only applicable to the event-based power analysis flow (power_method event_based) in the XP mode. The power_partition_count attribute must be specified with the power_use_physical_partition attribute.

Related Commands

- set_power

power_partition_twf

```
power_partition_twf {true | false}
```

Default: false

Data_type: bool

Specifies to read the input TWF data (flat or hierarchical TWF files), and generate block-level TWFs.This attribute must be specified when the block/partition level TWF files are not available.

Related Commands

- set_power

power_pin_based_twf

```
power_pin_based_twf {true | false}
```

Default: false

Data_type: bool

Specifies whether to generate pin-based TWF or net-based TWF.

- When set to `false`, a net-based TWF file is generated. Net-based TWF file is smaller in size and therefore results in faster processing.
- When set to `true`, a pin-based TWF file is generated. Pin-based TWF file is bigger in size and requires longer processing time, however, is more accurate because the slews numbers are pin-based.

Note: This attribute must be used with the `read_sdc` command.

Related Commands

- `set_power`

power_precision

`power_precision` *value*

Default: `8`

Data_type: int

Specifies precision of decimal range [1-8] to ensure consistent decimal places are displayed for each power component. This attribute is used for precision handling in `report_instance_power` and `report_power`.

Related Commands

- `report_power`

power_pre_simulation_empty_period

`power_pre_simulation_empty_period` *value*

Default: `""`

Data_type: string

Specifies to add an empty pre-simulation period before simulation during IR drop analysis. Power analysis is not done for the pre-simulation period and flat dynamic currents are created.

Related Commands

- `report_power`

power_pre_simulation_period

`power_pre_simulation_period` *value*

Default: `""`

Data_type: string

Specifies to set a pre-simulation period for IR drop analysis. Pre-simulation is used for generating power analysis reports and dynamic currents.

Related Commands

- `report_power`

power_pre_simulation_power_exclude_period

`power_pre_simulation_power_exclude_period` *value*

Default: `""`

Data_type: string

Specifies to exclude power analysis during the pre-simulation period used for IR drop analysis. Pre-simulation is used only to create dynamic currents and not used for power reports.

Related Commands

- report_power

power_quit_on_activity_coverage_threshold

```
power_quit_on_activity_coverage_threshold  threshold_value
```

Default: 1.0

Data_type: double

Specifies the activity (VCD/FSDB/TCF) coverage threshold value. This attribute allows you to control the acceptable activity coverage that you are expecting and enforce the software to quit when it does not meet the criteria. When you specify the threshold value, the software exits if the real activity coverage is below the threshold.

Related Commands

- report_power

power_read_rcdb

```
power_read_rcdb  {true | false}
```

Default: false

Data_type: bool

Specifies to write a SPEF file containing only the total C for signal nets and pass it to the Dynamic Power engine. Dynamic power analysis requires only the total C for signal nets. Therefore, instead of writing a large SPEF file (with distributed RC) from the RCDB, this attribute specifies to create and pass a SPEF file with only total C for all the signal nets. This significantly reduces the parasitic annotation time in the Dynamic Power engine.

Related Commands

- report_power

power_relax_arc_match

```
power_relax_arc_match  {true | false}
```

Default: true

Data_type: bool

Specifies to relax arc match requirements. This attribute is used only when the exact arc is not found in a .lib. When set to true, the internal energy is calculated as an average of multiple arcs with relaxed arc matching. If this attribute is set to false, relaxed arc search is not done and 0 internal energy is returned.

Related Commands

- report_power

power_report_black_boxes

```
power_report_black_boxes {true | false}
```

Default: false

Data_type: bool

Specifies to report cells that are used as black boxes.

A black box is a cell which is either missing from the liberty file or has no functionality defined in the liberty file.

Related Commands

- report_power

power_report_idle_instances

```
power_report_idle_instances {true | false}
```

Default: `false`

Data_type: bool

Specifies to generate a report (`idleinstance.rpt`) that lists all the idle or non-switching instances.

The following is a snippet of the `idleinstance.rpt` report:

```
DFF22/DFF    F
DFF42/DFF    F
...
```

Here, the first column is the instance name and the second column is the type of instance (`F` indicates flop and `C` indicates combinational).

Related Commands

- report_power

power_report_instance_switching_info

```
power_report_instance_switching_info {all | output_logic | none}
```

Default: `none`

Data_type: enum

Specifies to generate a report for combinational instance switched during state-propagation. In the output file, it contains the toggle time, instance name, input pin event, and output pin event for each switching event of the combination gate. The possible arguments of this attribute are:

- `all` - Specifies to report instance toggle information for both input and output pins. When this attribute is set to `all`, the switching of both the input and output pins of all the instances are reported in the `voltus_power.stateprop.switchinst` file and the switching of flops are reported in the `voltus_power.stateprop.switchsrc` file.
- `output_logic` - Specifies to report instance toggle information for the output pins only. When this attribute is set to `output_logic`, the switching of the output pins of all the instances are reported in the `voltus_power.stateprop.switchinst` file and the switching of flops are reported in the `voltus_power.stateprop.switchsrc` file.
- `none` - Disables generation of instance toggle report. This is the default option. When this attribute is set to `none`, only switching of flops are reported in the `voltus_power.stateprop.switchsrc` file. The file does not report switching of instances.

This attribute is used for generating switching details of state-propagation and should be specified only for debugging purpose.

The following is a snippet of the `voltus_power.stateprop.switchinst` report:

```
*-------------------------------------------------
*Switch Time   Instance Name   Pin Name   Logic
*-------------------------------------------------
*
   0.097        B11/BUFF         I        rise
   0.097        DFF11/DFF        Q        rise
...
```

Here,

- `Switch Time`: switching time of instances that can be either combinational or sequential
- `Instance Name`: instance name
- `Pin Name`: input/output pin of the instance
- `Logic`: present logic of the pin that can be either "rise" or "fall"

Related Commands

- report_power

power_report_instance_switching_list

```
power_report_instance_switching_list filename
```

Default:

Data_type: string

Specifies to generate a report (`voltus_power.stateprop.switchlist`) containing all the switching times for the specified instances when the `power_enable_state_propagation` attribute is set to `true`.

The format of the specified file is:

```
instname1
instname2
...
```

The following command is an example of the state-propagation-based dynamic vectorless flow, and also specifies to generate a report (`voltus_power.stateprop.switchlist`) containing all the switching times for the instances specified in the `inst_list` file:

```
set_db power_method dynamic_vectorless power_enable_state_propagation true power_report_instance_switching_list inst_list
```

The following is the format of the `voltus_power.stateprop.switchlist` report:

```
*-----------------------------------------------------------------------------------------
*Instance Name   Pin Name   Pin Direction   Frequency   Switch Time
*-----------------------------------------------------------------------------------------
*
```

Here,

- `Instance Name`: instance name that can be either combinational or sequential
- `Pin Name`: name of the input/output pin of the instance
- `Pin Direction`: indicates whether the pin is an INPUT (I) or OUTPUT (O) pin
- `Frequency`: clock frequency
- `Switch Time`: gives the switching pattern for the instance, that is, <*time1*> <*logic*> <*time2*> <*logic*>. The logic of the pin can be "`r`" or "`f`". `r` indicates rise and `f` indicates fall.

The following is a snippet of the `voltus_power.stateprop.switchlist` report:

```
*-----------------------------------------------------------------------------------------
*Instance Name   Pin Name   Pin Direction   Frequency   Switch Time
*-----------------------------------------------------------------------------------------
*
   DFF11/DFF        QN        O              100       0.106 f 10.107 r
   DFF11/DFF        Q         O              100       0.097 r 10.099 f
   DFF11/DFF        CP        I              100       0.059 r 5.056 f 10.059 r 15.056 ...
...
```

Related Commands

- `report_power`

power_report_library_usage

```
power_report_library_usage {true | false}
```

Default: `false`

Data_type: bool

Specifies to report library usage statistics in a text report. The report contains information such as, cell name, library format used (CCSP/ECSM/NLPM) for each cell, library name, and so on. This attribute allows you to perform library integrity checking for debugging purpose.

The format of the text report with sample content is given below:

Example:

```
#cell_name  net_name    op_voltage  lib_voltage   lib_format     missing_tables   library_name
```

```
INV         VDD,VDDA       0.9        1.0          NLDM           leakage,timing      A
INV         VDDC           0.75       0.75         CCSP,NLDM      leakage             B
```

Related Commands

- report_power

power_report_missing_bulk_connectivity

```
power_report_missing_bulk_connectivity {true | false}
```

Default: `false`

Data_type: bool

Specifies to report missing bulk pins in the missing input report (`power_report_missing_input`).

Related Commands

- report_power

power_report_missing_input

```
power_report_missing_input {true | false}
```

Default: `false`

Data_type: bool

Specifies to report missing netlist information in input files, such as library, LEF/DEF, PGV, SPEF, TWF, and missing logical connectivity for a cell instance (PGNET).

The name of the report file is: `<prefix>_missingdata.<spef|lefdef|lib|pgv|twf|pgnet>`

Related Commands

- report_power

power_report_missing_nets

```
power_report_missing_nets {true | false}
```

Default: `false`

Data_type: bool

Controls reporting of missing nets in both static and dynamic power analysis.

This attribute controls reporting for TWF and activity files in static analysis, and controls reporting for TWF/SDC/SPEF and activity files in dynamic analysis.

Related Commands

- report_power

power_report_scan_chain_statistics

```
power_report_scan_chain_statistics {true | false}
```

Default: `false`

Data_type: bool

Specifies to generate a report file called `voltus_power.scanchain.stat` during scan mode analysis that contains scan chain statistics.

The report has 2 parts:

- **Global Statistics**: this section has the cumulative statistics of all scan chains in the design
- **Chain Statistics**: this section has the statistics for each scan chain in the design

The Global and Chain Statistics sections have the following information:

- `Total`: total instances specified in the control file or in a scan chain
- `AssignedFF`: flip-flops defined in the scan control file which exist in the design
- `SwitchedFF`: flip-flops that have switching event at the output
- `Non-ScanFF`: flip-flops that have no scan-out pin
- `DuplicatedFF`: flip-flops that are specified in multiple chains
- `StateConflictFF`: flip-flops that have conflicting state assignment

Related Commands

- `report_power`

power_report_statistics

```
power_report_statistics {true | false}
```

Default: `false`

Data_type: bool

Reports a set of statistics on the instance power, instance power density, clock power, or transition density. This attribute is useful in viewing and debugging user defined activity factors.

Related Commands

- `report_power`

power_report_time_display_fraction_digits

```
power_report_time_display_fraction_digits value
```

Default: `-1`

Data_type: int

Sets the number of fractional digits or decimal places displayed for the instance switching time values in the `voltus_power.stateprop.switchinst` file. This attribute displays fractional digits that show the exact time difference if there are two events (rise/fall) of an instance switching at the same time.

Related Commands

- `report_power`

power_report_twf_attributes

```
power_report_twf_attributes {summary| detailed}
```

Default:`""`

Data_type: string

Controls the generation of the TWF attribute report based on the TWF attributes that are specified using the `set_twf_attribute` command.

The possible arguments are:

- `summary` - generates a summarized static timing analysis report "`voltus_power.twf_attribute.summary`" that displays the number of user-defined TWF attributes (delay, slew, and slack values for both the rise and fall transitions) and the number of annotated TWF attributes for dynamic power analysis. The following is a snippet of the report:

```
STATISTICS:
                    User Defined            Annotated               Annotated/Total(%)
shift_arrival_time   5                       4                       80
rise_slew            2                       2                       100
fall_slew            2                       2                       100
rise_delay           5                       4                       80
fall_delay           5                       4                       80
rise_slack           2                       2                       100
fall_slack           2                       2                       100
```

- `detailed` - generates a detailed static timing analysis report " `voltus_power.twf_attribute.detailed`" that includes the summary report, and a detailed report of the rise and fall transitions for the TWF attributes of each annotated instance/pin. It also includes the shift arrival time report.

  This argument also allows you to generate a report `voltus_power.twf.rpt` containing the TWF statistics to highlight clock-frequency and delay used for the static and dynamic power calculation.

  The TWF statistics report is generated only if:

  - the `read_twf` command is specified before specifying the `power` category attributes.
  - the `power_method static power_static_netlist def` or `power_method dynamic_vectorless | dynamic_vectorbased` command is specified.

  The format of the `voltus_power.twf.rpt` report is:

```
# Voltus TWF Statistics
# Units: Freq (Hz), Delay (second), Slew (second)
# TWF File List:
#     TWF_file1
#     TWF_fileN

USER DEFINED TWF ATTRIBUTE STATISTICS
<Clock Freq> <Min Rise_Delay> <Rise Delay> <Max Rise Delay> <Min Fall Delay> <Fall Delay> <Max Fall Delay> <Net>
```

  The `USER DEFINED TWF ATTRIBUTE STATISTICS` section is included in the report only if the TWF attributes are defined using the `set_twf_attribute` command.

Example :

#check all instances of inst_C1 get shifted by 3ns

```
set_twf_attribute -pin inst_C1 -shift_arrival_time 3
```

#check all instances of inst_D1 get shifted by 3ns

```
set_twf_attribute -net inst_D1 -shift_arrival_time 3

set_db power_method dynamic_vectorless ; set_db power_write_db false ; set_db power_disable_static false ; set_db
power_enable_state_propagation true; set_db power_current_generation_method peak ; set_db power_grid_libraries /../library_pv.cl ; set_db
power_report_twf_attributes summary
```

Related Commands

- `report_power`

power_reuse_flop_ranking_data

```
power_reuse_flop_ranking_data directoryname
```

Default: `""`

Data_type: string

Specifies the name of the directory where the flop ranking data to be reused is placed.

Related Commands

- `report_power`

power_reuse_flop_ranking_data_hier

```
power_reuse_flop_ranking_data_hier blockname
```

Default: `""`

Data_type: string

Allows you to choose the block/hierarchy whose ranking data is to be used for the block-level analysis. This attribute is used when there are multiple instantiations of the same DEF block.

Related Commands

- report_power

power_scale_to_sdc_clock_frequency

```
power_scale_to_sdc_clock_frequency {true | false}
```

Default: false

Data_type: bool

Specifies to enable the VCD clock frequency to be scaled up to the SDC clock frequency. This allows you to match the VCD clock to the SDC clock to run analysis. By default, the clock frequency is scaled based on the activity file. When you set the power_scale_to_sdc_clock_frequency attribute to true, the power report and current waveforms apply the after-scaled clock frequency.

The default behavior for clock frequency scaling in the RTL and Gate level VCD flows is different, as described below:

- **RTL VCD** - By default, the clock frequency is scaled to the SDC clock frequency (power_scale_to_sdc_clock_frequency true) when the RTL VCD flow is enabled (power_enable_rtl_dynamic_vector_based true).
  You can disable clock frequency scaling by setting the power_scale_to_sdc_clock_frequency attribute to false.
- **Gate VCD** - By default, scaling of the clock frequency is disabled (power_scale_to_sdc_clock_frequency false) .
  You can enable the VCD clock frequency to be scaled up to the SDC clock frequency by setting the power_scale_to_sdc_clock_frequency attribute to true. This works only when power_use_zero_delay_vector_file is set to true, which means that the gate VCD is a zero-delay gate VCD.

Related Commands

- report_power

power_scan_chain_activity

```
power_scan_chain_activity {design_name  chain_name  activity_number} | {vectorless  activity_number}
```

Default: ""

Data_type: string

Specifies to either provide vectorless activity for all scan chains or provide a specific activity for a specific chain. This attribute allows you to specify vectorless or user-defined activity to scan chains.

The following are the two use models:

- Generate the scan control file for a specific chain:
  ```
  set_db power_scan_chain_activity {design_name  chain_name  activity_number}
  ```
- Generate the scan control file for all scan chains:
  ```
  set_db power_scan_chain_activity {vectorless  activity_number}
  ```

The power_scan_chain_activity attribute is supported only in the XP mode.

Related Commands

- report_power

power_scan_chain_name_pattern

```
power_scan_chain_name_pattern {design_name  chain_name  pattern}
```

Default: ""

Data_type: string

Specifies to automatically generate a scan-chain control file from a scan-chain based DEF file for the shift mode scan-chain analysis. This attribute allows you to create a control file for a large-sized DEF containing long chains or for multiple-DEF files. The generated scan-chain control file is saved in the power output directory.

The use model of the attribute is:

- power_scan_chain_name_pattern {all pattern} - generate scan control file for all scan chains.

Example: `power_scan_chain_name_pattern {all 11010}`

- `power_scan_chain_name_pattern {design_name chain_name pattern}` - generate scan control file for a specific chain.
  Example: `power_scan_chain_name_pattern {counter AutoChain_1_seg1_clk_rising 10010}`

Using this attribute, you can generate the control file for both flat and hierarchical DEFs. For more information, refer to Scan Control File Format.

Related Commands

- `report_power`

power_scan_control_file

`power_scan_control_file` *`filename`*

Default: `""`

Data_type: string

Specifies the scan control file name required to run the Scan Mode Analysis flow.

The scan control file contains details about the scan analysis mode used for testing. For more information, refer to Scan Control File Format.

Related Commands

- `report_power`

power_scan_multi_bit_flop_chain_type

`power_scan_multi_bit_flop_chain_type` *`type`*

Default: `liberty`

Data_type: string

Specifies the type of multi-bit scan flip-flop.

- `liberty` - specifies to use the syntax in Liberty to decide the multi-bit flip-flop type.
- `serial` - specifies to pick the first flop as the starting flop, and treat the scan chain as serial, regardless of .lib.
- `parallel` - specifies to treat the scan chain as parallel, regardless of .lib.

Related Commands

- `report_power`

power_settling_buffer

`power_settling_buffer` *`value`*

Default: `-1.0`

Data_type: string

Specifies the settling buffer time (in ps unit) between multiple windows of a VCD file. This attribute is used when you specify multiple pairs of the start time and end time for the non-overlapping multiple windows specified in an activity file (using the `read_activity_file` command).

Multiple current waveforms representing multiple VCD windows are merged into a single continuous current waveform. To capture the waveform for signals at the edge of user windows, a buffer time is added between the windows being stitched. This buffer time ensures that the stitched waveform is contiguous. The generated waveform hence has the duration of user windows and the intervening settling buffers. The default settling buffer period is 400ps. You must specify the buffer value without unit.

Related Commands

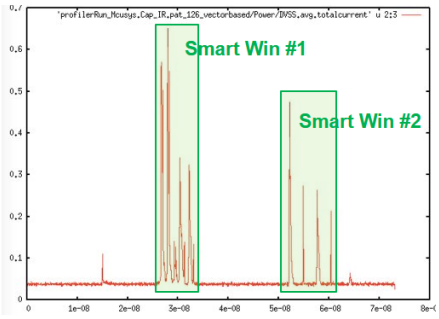
- `report_power`

power_smart_window

```
power_smart_window {true | false}
```

Default: `false`

Specifies to enable the smart window feature to extract peak power scenarios from a vector-based activity file. A smart window is a variable size window to capture maximum regions of high power dissipation. An activity file can have multiple variable-sized regions that are ranked in the descending order of power metric.



The use model of the `power_smart_window` attribute is:

```
set_db power_smart_window true
set_db power_worst_step_size value
set_db power_worst_window_type [ power | activity | delta_power | delta_activity | ir | critical_inst_power ]
```

Related Commands

- `report_power`

power_split_bus_power

```
power_split_bus_power {true | false}
```

Default: `false`

Data_type: bool

A value of `false` applies the internal power number to each individual bit.

A value of `true` divides the internal power number by bus width, before applying it to the bus bit.

Related Commands

- `report_power`

power_start_time_alignment

```
power_start_time_alignment {true | false}
```

Default: `true`

Data_type: bool

Specifies to align all the signals of the activity file at time zero. When set to `true`, the simulation period is `max(endtime - starttime)`. When it is `false`, the simulation period is `max(endtime)- min(startime)`.

Note: This attribute is not supported in the vector profiling flow. In the vector profiling flow, the start time will always be the `min(starttime)` and the end time will always be `max(endtime)`

Related Commands

- `report_power`

power_state_dependent_leakage

```
power_state_dependent_leakage {true | false}
```

Default: `true`

Data_type: bool

When set to false, power analysis performs state independent leakage power calculation.

Related Commands

- report_power

power_stateprop_ignore_unannot_pins

```
power_stateprop_ignore_unannot_pins {true | false}
```

Default: `false`

Data_type: bool

Specifies to ignore unannotated pins in the functional logic of output pins for the RTL propagation flow. This attribute is set to `true` if there are unannotated input pins for instances and such pins are used in the functional logic of output pins that are propagated. For example output pin Q is propagated with functional logic A & B where A is unannotated and might cause wrong propagation for the Q pin.

Related Commands

- report_power

power_static_multi_mode_scenario_file

```
power_static_multi_mode_scenario_file  filename
```

Default: `""`

Data_type: string

Specifies a file with information on instance scaling scenario for the static multi-mode IR analysis flow. When this parameter is specified, Power Analysis creates the multi-mode static current file that is used during rail analysis. The format of the input file is:

```
INST_NAME       PIN_NAME        SCALE_FACTOR_1       …..   SCALE_FACTOR_N
```

Here, the first column indicates the macro instance name, the second column indicates the pin name, followed by multiple current scaling factors. Each column of the scaling factor is referred to as a scenario. You must specify positive values only for scale factors.

The following is a snippet from the specified input file:

```
top/macroA     VDD     1     0.5     0.5     1
top/macroA     VDDM    0.5   0.5     0.5     1
top/macroB     VDD     0.5   1       0.5     0.5
```

If scale factors are not specified for certain scenarios, the tool will automatically use the instance's last stored scale factor value to complete current file generation.

In the following example, scale factor value `0.45` will be used in the 3rd and 4th scenario for instanceC, and scale factor value `0.35` will be used in the 4th scenario for instanceD:

```
instanceA VDD 0.45 0.55 0.65 0.31
instanceB VDD 0.31 0.5 0.67 0.21
instanceC VDD 0.25 0.45
instanceD VDD 0.31 0.25 0.35
```

Related Commands

- report_power

power_static_netlist

```
power_static_netlist  {verilog | def}
```

Default: `verilog`

Data_type: string

Specifies to perform static power analysis using a Verilog or DEF only neltist.

By default, the static power analysis is performed using the Verilog netlist as the primary netlist. Using this attribute, you can specify DEF when a DEF netlist with logical connectivity is available, and use it as primary and not consider the Verilog netlist.

This attribute is also enabled when `set_rail_analysis_config -report_power_in_parallel` is `true` for static power computation.

Related Commands

- `report_power`

power_switching_power_on_rise_only

```
power_switching_power_on_rise_only {true | false}
```

Default: `false`

Data_type: bool

Specifies to perform switching power analysis only for rise toggles.

Related Commands

- `report_power`

power_thermal_input_file

```
power_thermal_input_file  file
```

Default: `""`

Data_type: string

Specifies to read a power map file to perform thermal analysis. Thermal input file contains die area divided into tiles, and temperature value for each tile. This temperature of each tile is used to calculate power of all the instances that fall in that tile's bounding box.

The format of the thermal input file is:

```
DIE_TEMPERATURE_MAP
DIE_AREA   xmin(um) ymin(um) xmax(um) ymax(um) nx(int) ny(int)

NUMBER_OF_LAYERS    nLayers(int)
LAYER    layer_name(string)
TEMPERATURE

XCOOR    x0(0)    x0(1)    x0(2)... ...x0(nx-1)
YCOOR    y0(0)    y0(1)    y0(2)... ...y0(ny-1)

T0(0,0)    T0(1,0)    ...    T0(nx-1,0)
T0(0,1)    T0(1,1)    ...    T0(nx-1,1)

:
T0(0,ny-1)    T0(1,ny-1)          T0(nx-1,ny-1)
END_TEMPERATURE
END_LAYER
END_DIE_TEMPERATURE_MAP
```

where,

- `DIE_AREA` specifies the coordinates of the four points of die outline as well as the tile number in X and Y direction.
- `NUMBER_OF_LAYERS` is the number of layers in each die instance
- `LAYER` specifies the LEF layer name
- XCOOR and YCOOR are the coordinates of each tile
- `T0(0,0)` …. is the temperature value assigned to a tile at a specific coordinate

Related Commands

- report_power

power_thermal_leakage_temperature_scale_table_file

```
power_thermal_leakage_temperature_scale_table_file file
```

Default: ""

Specifies a file containing the user-defined temperature values and temperature scale factors for each instance. This attribute allows you to use the current file (.ptiavg) with re-adjusted leakage power values to perform thermal-aware IR drop analysis. The power_thermal_leakage_temperature_scale_table_file attribute must be specified with the power_thermal_input_file attribute to import the temperature map file generated by Celsius.

When the new attribute is specified, the generated power will include the scaled leakage power values.

The following is the format of the temperature-aware leakage scaling table:

```
#Type    Instance_Name        temp1             temp2              temp3              temp4
(Type: Lib/Cell/inst)

inst     inst_name1     scalefactor@temp1   scalefactor@temp2  scalefactor@temp3   scalefactor@temp4
inst     inst_name2     scalefactor@temp1   scalefactor@temp2  scalefactor@temp3   scalefactor@temp4
```

File Format Example:

```
#Type     Name        25    50    125    150

inst     ring/b213    1     2     5      10
inst     ring/b212    1     2     5      10
inst     ring/b211    1     2     5      10
inst     ring/b210    1     2     5      10
```

Related Commands

- report_power

power_to_x_transition_factor

```
power_to_x_transition_factor value
```

Default: 0.5

Data_type: double

Specifies how 0/1 to X transitions are counted.

Related Commands

- set_power

power_to_z_transition_factor

```
power_to_z_transition_factor value
```

Default: 0.25

Data_type: double

Specifies how 0/1 to Z transitions are counted.

Related Commands

- set_power

power_transition_factor_based_duty

```
power_transition_factor_based_duty {true | false}
```

Default: `false`

Data_type: bool

Specifies that duty for X and Z toggles is calculated using the X and Z transition factor of toggle specified using the following attributes:

- `power_from_x_transition_factor`
- `power_from_z_transition_factor`
- `power_to_x_transition_factor`
- `power_to_z_transition_factor`

The default value of `false` specifies to use the duty value given with the `set_default_switching_activity`/`set_switching_activity -duty` parameter.

Related Commands

- `report_power`

power_transition_time_method

```
power_transition_time_method  {min | avg | max}
```

Default: The default value is:

- `min` for static power analysis using the Verilog netlist (`-static_netlist verilog`).
- `avg` for static power analysis using the DEF netlist (`-static_netlist def`) and dynamic power analysis.

Data_type: string

Specifies the minimum, maximum, or average transition time method that will be used with the integrated timer or the external TWF.

Related Commands

- `report_power`

power_twf_delay_annotation

```
power_twf_delay_annotation {min | avg | max}
```

Default: `avg`

Data_type: string

Allows you to choose between min, max, or avg arrival times from the timing window file (TWF). The arrival time is annotated to the vectors processed from the RTL VCD/FSDB Zero-Delay flows.

Related Commands

- `report_power`

power_twf_load_cap

```
power_twf_load_cap {min | avg | max}
```

Default: `max`

Data_type: string

Allows you to select minimum, maximum, or average value of TWF external load or capacitance for power calculation.

Related Commands

- `report_power`

power_use_cell_leakage_power_density

```
power_use_cell_leakage_power_density {true | false}
```

Default: `true`

Data_type: bool

Specifies to use library leakage density times area (`default_leakage_power_density`), if defined, instead of the library's default cell leakage power (`default_cell_leakage_power`). This is used when leakage power for cells are missing or not defined.

Related Commands

- `report_power`

power_use_fastest_clock_for_dynamic_scheduling

```
power_use_fastest_clock_for_dynamic_scheduling {true | false}
```

Default: `false`

Data_type: bool

Specifies to use only the respective fastest clock associated with each net or pin to schedule events for a given simulation period.

Related Commands

- `report_power`

power_use_lef_for_missing_cells

```
power_use_lef_for_missing_cells {true | false}
```

Default: `false`

Data_type: bool

Allows a combination of LEF and PGV in the dynamic analysis flow. If there are cells that are common to LEF and PGV, the PGV cell will get the precedence.

Related Commands

- `report_power`

power_use_physical_partition

```
power_use_physical_partition {true | false}
```

Default: `false`

Data_type: bool

Enables tile or area-based partitioning of a design for distributed processing. When this attribute is specified, the top-level design is divided into physical tiles, which are then grouped to ensure that each group or partition has the same instance count. Each partition is considered and processed as a separate job.

This attribute is only applicable to the event-based power analysis flow (`power_method event_based`) in the XP mode. The `power_use_physical_partition` attribute must be specified with the `power_partition_count` attribute.

Example:

The following command specifies to enable area-based partitioning, and distribute the event-based analysis into four jobs:

```
set_db power_method event_based power_enable_xp true power_use_physical_partition true power_partition_count 4
report_power -distribute
```

Related Commands

- `report_power`

power_worst_case_vector_activity

```
power_worst_case_vector_activity {true | false}
```

Default: `false`

Data_type: bool

Specifies to use the worst activity value when multiple vectors are specified in the static power calculation flow.

Related Commands

- `report_power`

power_worst_step_size

```
power_worst_step_size value
```

Default: `Total simulation duration/100`

Data_type: string

Specifies the step size or resolution for vector profiling. Activity/ Power profiling is carried out for every step size that you specify.

When this attribute is used with `power_worst_window_size`, the specified step size is the resolution at which window sliding is performed. The supported time units are s, ms, us, ns, and ps. If no explicit time unit is specified, the default ns will be applied.

Related Commands

- `report_power`

power_worst_window_count

```
power_worst_window_count value
```

Default: 1 (implements single worst window)

Data_type: integer

Specifies the number of worst windows from the last vector profile run to be used for dynamic power analysis. It provides n top windows with maximum activity/power/IR. This attribute is used when you need to combine dynamic power analysis and vector profiling for multiple worst-case windows in a single run.

Related Commands

- `report_power`

power_worst_window_coverage

```
power_worst_window_coverage { top | tile }
```

Default: `top`

Specifies how the worst window type is applied for multi-window analysis in the vector profiling flow. The following are the possible options of this attribute:

- `top`: worst window selection for the full design
- `tile`: interval selection as per the worst tile power

Use Model:

```
set_db power_method event_based power_worst_window_coverage tile
report_power -out_file power.rpt -power_density_tiles 100
```

This flow should be run with `power_method event_based`.

When the `power_worst_window_coverage tile` attribute is specified, you should include any one of the following parameters to specify the total number of tiles:

- `report_power -power_density_tiles_size " 15 15 "`
- `report_power -power_density_tiles_row_column " 2 2 "`
- `report_power -power_density_tiles 100`

The software generates the `power.rpt.density` report that has the power density tiles profile report with an additional section "Intervals covering unique tiles having worst power density".

Related Commands

- `report_power`

power_worst_window_reports

```
power_worst_window_reports {full | worst | both}
```

Default: `full`

Data_type: string

Specifies to generate static power analysis reports for the specified worst window type. The possible arguments are:

- `full`: creates the static power report for the full simulation window specified with the `-start`/`-end` parameters of the `read_activity_file` command.
- `worst`: creates the static power report for the worst window only.
- `both`: creates the static power report for both the full simulation and worst windows.

The worst window report is created with '**.1**' added in the file name, as shown below:

- Static reports for full simulation: `power.rpt.avgpwr`, `power.rpt.instpwr`
- Static reports for worst window: `power.rpt.1.avgpwr`, `power.rpt.1.instpwr`

Related Commands

- `report_power`

power_worst_window_size

```
power_worst_window_size value
```

Default: `""`

Data_type: string

Specifies the duration of dynamic analysis. This attribute is used for sliding window vector profiling that involves sliding the profiling windows by using a very small resolution value to get the window with the worst current profile. `power_worst_window_size` must be specified with `power_worst_step_size`.

The supported time units are s, ms, us, ns, and ps. If no explicit time unit is specified, the default ns will be applied.

Related Commands

- `report_power`

power_worst_window_type

```
power_worst_window_type  {power | activity | delta_power | delta_activity | ir | critical_inst_power}
```

Default: `power`

Data_type: string

Specifies the window type to be used for multi-window analysis. The possible arguments are:

- `power` - worst instance power
- `activity` - worst instance activity
- `delta_power` - worst dp/dt (rate of change of power)
- `delta_activity` - worst da/dt (rate of change of activity)
- `ir` - worst IR
- `critical_inst_power` - worst power for critical/IR-sensitive instance (worst timing and IR)

Related Commands

- report_power

power_use_zero_delay_vector_file

```
power_use_zero_delay_vector_file {true | false}
```

Default: false

Data_type: bool

Enables zero delay mode in vector-based dynamic analysis to avoid pessimism in current estimation. When set to true, the software reads the cycle accurate (zero delay) VCD/FSDB file and adds the timing/delay information from the Timing Analyzer or external TWF file to the VCD/FSDB file.

Related Commands

- report_power

power_vector_based_multithread

```
power_vector_based_multithread {true | false}
```

Default: true

Data_type: bool

Specifies to enable multi-threading for the VCD/FSDB based dynamic vector-based flows.

To enable multi-threading in the vector-based flows, set this attribute to true.

Related Commands

- report_power

power_vector_profile_mode

```
power_vector_profile_mode {activity | event_based | power_density | transient}
```

Default: event_based

Data_type: string

Specifies the method used to perform vector profiling. The possible arguments are:

- activity - performs activity profiling. This parameter generates only the worst activity report, and does not generate the average power report. The name of the file is as specified using the report_power -out_file parameter.
- event_based - generates the average power report. This is the average power of all events in the user-specified window. This report includes a summary based on the power components, cell type, power rail, and clock network. The worst window report is captured in a file specified with the report_power -out_file parameter, and the report is suffixed by '.avgpower'. When -out_file is not specified, the vectorprofile.report.avgpower report file is created.
- power_density - performs area or power density profiling. For this method, power is reported for each tile of the design, where design can be considered to be physically divided into equal area tiles. By default, a design is divided in 10x10 tiles.
- transient - generates the event-based average power report and current files for the worst power window.

Related Commands

- report_power

power_write_bbox

```
power_write_bbox  {true | false}
```

Default: false

Data_type: bool

Specifies to save the boundary box of each instance, its placement, and rotation at the top level. When set to `true`, generates the power database with each instance boundary box, placement, and rotation information at the top level.

Related Commands

- `report_power`

power_write_db

```
power_write_db {true | false}
```

Default: `false`

Data_type: bool

A value of `true` creates a binary database of power results.

Related Commands

- `report_power`

power_write_default_pti_files

```
power_write_default_pti_files {true | false}
```

Default: `true`

Data_type: bool

For instances that are not hooked to any power/ground rail, the current for these instances are reflected in the default static current files:

- `static_default_ground_rail.ptiavg`
- `static_default_power_rail.ptiavg`

Related Commands

- `report_power`

power_write_dynamic_currents

```
power_write_dynamic_currents {true | false}
```

Default: `false`

Data_type: bool

Specifies to create dynamic current files either for full simulation duration or for user-specified duration. This attribute is only applicable to the event-based power analysis flow (`power_method event_based`).

Example:

```
set_db power_method event_based power_write_dynamic_currents true power_enable_state_propagation true
read_activity_file sim.vcd -format vcd
report_power -out_file power.rpt
```

Related Commands

- `report_power`

power_write_gui_db

```
power_write_gui_db {true | false}
```

Default: `false`

Data_type: bool

Specifies to create a binary database of power results in the XP mode. When set to `true`, the tool creates sliced power database file for each rail partition and saves it to the `gui.db` sub-directory under the power output directory (example: `power_dir/part_*/gui.db/rail_part_*/power.x.db` (x = 1, 2, ..., 14). The generated power database allows you to perform GUI interactive debugging using the *Power & Rail Plots* and *Power Debug* forms.

Related Commands

- `report_power`

power_write_profiling_db

```
power_write_profiling_db {true | false}
```

Default: `false`

Data_type: bool

Writes out profiling databases which can later be viewed as histograms using the SimVision interface. You can specify this parameter during both the activity and power vector profiling. The profiling databases support instance level and power/ground net histograms. It creates an output file name with the `*.trn` extension.

Related Commands

- `report_power`

power_write_simulation_db

```
power_write_simulation_db  {true | false}
```

Default: `false`

Data_type: bool

Specifies to save functional waveforms for each instance when parsing an activity file. When set to `true`, creates SimVision waveform database (SHM database) for toggle simulation used during power analysis, which can later be viewed as histograms using the SimVision interface.

The simulation database (`shmdb.trn`) is generated in the power analysis output directory.

Related Commands

- `report_power`

power_write_static_currents

```
power_write_static_currents {true | false}
```

Default: `false`

Data_type: bool

A value of `true` generates the current data files per net.

Note: When you perform static power calculation in the dynamic power analysis flow, you can use either the `power_disable_static` or `power_write_static_currents` attribute to enable static power calculation and write static current file. If only `power_disable_static` is set to `false` and `power_write_static_currents` is not set, the tool automatically sets `power_write_static_currents` to `true`. Similarly, if `power_write_static_currents` is set to `true`, and `power_disable_static` is not set, the tool automatically sets `power_disable_static` to `false`.

Related Commands

- `report_power`

power_x_count_transition_using_3_states

```
power_x_count_transition_using_3_states {true | false}
```

Default: `false`

Data_type: bool

Specifies to calculate the toggle rate using 3 states, pre-previous, previous and current states.

Related Commands

- `report_power`

power_x_transition_factor

```
power_x_transition_factor value
```

Default: `0.5`

Data_type: double

When using VCD/FSDB, one can specify how transitions to and from X are counted. These transitions are defined as shown in Table 1 below. Transitions to and from the X state will be treated as full transitions multiplied by the specified factor.

Note: 0 -> X -> 1 or 1 -> X -> 0 count as full transitions and they are not controlled by the attribute.

Related Commands

- `report_power`

power_z_transition_factor

```
power_z_transition_factor value
```

Default: `0.25`

Data_type: double

When using VCD/FSDB, one can specify how transitions to and from Z are counted. These transitions are defined as shown in Table 1 below. Transitions to and from the Z state will be treated as full transitions multiplied by the specified factor.

Note: 0 -> XZ-> 1 or 1 -> Z-> 0 count as full transitions and they are not controlled by the `attribute`.

Related Commands

- `report_power`

Table **1: Transitions from/to X and Z values and**

Transition	Definition	Default
to/from X	[0 or 1 or Z] <-> X	0.5
to/from Z	[0 or 1] <-> Z	0.25

power_zero_delay_vector_toggle_shift

```
power_zero_delay_vector_toggle_shift value
```

Default: `0`

Data_type: string

Specifies to shift the current waveform in the zero-delay VCD flow. This attribute is useful in removing the idle period for the current waveform result with a long idle period (small flat current) before the first high current peak because of delay annotation. The attribute allows to offset the idle period from the delay values.

The default value is 0ns. The supported time units are s, ms, us, ns, and ps. The default unit is ns. You can specify a negative or a positive value. A positive value means right shift in the timeline or add more delay to the original arrival time, and a negative value means left shift in the timeline or trim the unwanted idle period.

The `power_zero_delay_vector_toggle_shift` attribute enables you to make a more realistic current waveform that can be easily aligned with the other current waveforms.

Note: This attribute is similar to the `set_twf_attribute -shift_arrival_time` parameter, with the only difference that `power_zero_delay_vector_toggle_shift` supports both positive and negative values while `-shift_arrival_time` supports only a positive value.

Related Commands

- `report_power`

Scan Control File Format

The use model for the Scan Mode Analysis flow is:

```
set_db power_method dynamic_vectorbased power_scan_control_file filename
```

The format of the scan control file is:

```
MODE = [ SHIFT | VECTORLESS]
VECTORLESS_GLOBAL_ACTIVITY = <factor> [Between 0 - 1]
<chain_name> <flop_inst> <initial_state> <master_cell_name> <scale> <mode> <scan_in_pin> <scan_out_pin> <scan_en_pin>
```

Note that <`chain_name`> is optional, <`initial_state`> is optional in vector-less mode, and <`master_cell_name`>, <`scale`>, <`mode`>, <`scan_in_pin`>, <`scan_out_pin`>, <`scan_en_pin`> are optional and only needed for the first internal scan flop inside one macro.

Where,

- `Mode` specifies the mode used for scan design analysis. The two modes are: `SHIFT` and `VECTORLESS`.
- `VECTORLESS_GLOBAL_ACTIVITY` specifies the activity specification used in the `VECTORLESS` mode.
- `chain_name` is the name of the scan chain.
- `initial_state` is the initial state of all scan flops.
- `master_cell_name` is the name of the macro.
- `scale` is the scaling factor applied to the current.
- `mode` is mode name of the scan mode in the PGV view.
- `scan_in_pin` is the macro's scan_in pin name.
- `scan_out_pin` is the macro's scan_out pin name.
- `scan_en_pin` is the macro's scan_enable pin name.

Related Information

- `report_power`
- `set_power_output_dir`
- "Static Power, IRdrop and EM Analysis" in the *Voltus Stylus Common UI User Guide*
- "Dynamic Power and IRDrop Analysis" in the *Voltus Stylus Common UI User Guide*

propagate_activity

```
propagate_activity
[-set_net_frequency {true | false}]
```

Propagates the activity file in the database after it is read in using the `read_activity_file` command. This command propagates the activity for all primary

inputs, nets, and other devices in the design whose activity has not been previously defined through user attributes.

`propagate_activity` can be used to propagate activities in the design and check for activity annotation before running power calculation. This is not a mandatory step as execution of `report_power` will also do the same and then do power calculation based on the propagated activities.

Parameters

`-set_net_frequency {true | false}`
Specifies that activity from the generated TCF file needs to be used to capture net frequency for signal EM analysis. When this parameter is set, you do not have to read the TCF file to set net frequency. *Data_type*: enum, optional

Example

- The following commands read and propagate the activity file in the database:

  ```
  read_activity_file
  propagate_activity
  ```

Related Information

- `read_activity_file`
- "Dynamic Power and IRDrop Analysis" in the *Voltus Stylus Common UI User Guide*

query_power_data

```
query_power_data
[-help]
[<pti_files <file1[file2] ...>>]
[-average_current]
[-clock_average]
[-write_pwl]
[-frame <string>]
[-insts <inst_list>]
[-inst_list_file filename]
[-list]
[-out_dir <string>]
[-peak_current]
[-peak_time]
[-pin]
[-time_steps]
[-verbose]
[-total_current <input_current_file> ]
[-output_tap_name <tap_name>]
```

The `query_power_data` command allows you to query the binary current files (.ptiavg/.ptipeak) that Power Calculation/Rail Analysis creates. It reads the tap current data in the current files and reports various information for debugging.

Parameters

`-help`	Outputs the command usage and a brief description about the command parameters.
`<pti_files <file1[file2] ...>>`	
	Specifies the names of the VDD and VSS instance voltage pti files. *Data_type*: string, optional
`-average_current`	
	Specifies to write the average current of all instances. *Data_type*: bool, optional
`-clock_average`	
	Specifies to write the clock based current report. *Data_type*: bool, optional
`-write_pwl`	
	Specifies to write the tap current in the PWL format. You can use this parameter to view the contents of the binary data file. *Data_type*: bool, optional
`-frame <string>`	
	Specifies intervals for the clock_average report. *Data_type*: string, optional
`-insts <inst_list>`	

	Specify the list of instances to be dumped in the ASCII format, one instance per line. If not specified, it prints all instances. The `-insts` parameter is specified with the `-write_pwl`, `-peak_current`, `-peak_time`, and`-average_current` parameters. *Data_type*: string, optional
`-inst_list_file` *filename*	
	Specifies a file with the list of instances to be dumped in the ASCII format. *Data_type*: string, optional
`-list`	
	Specifies to write the tap names from the input file. *Data_type*: bool, optional
`-out_dir` <*string*>	
	Specifies the name of the output file. *Data_type*: string, optional
`-output_tap_name` <*tap_name*>	
	Specifies the name of the output tap in the new current file, which is created using the `-total_current` parameter. *Data_type*: string, optional
`-peak_current`	
	Specifies to write the peak current of all instances. *Data_type*: bool, optional
`-peak_time`	
	Specifies to write the peak current time of all instances. *Data_type*: bool, optional
`-pin`	
	Specifies to print the pin names with the instance names. The `-pin` parameter is specified with the `-write_pwl` and `-list` parameters. *Data_type*: bool, optional
`-verbose`	
	Specifies to print the report to standard output. *Data_type*: bool, optional
`-time_steps`	
	Specifies to summarize the minimum current, maximum current, average current, and total current for each interval in the data file. You can use this data to estimate which interval in a transient response might contain the worst-case data. *Data_type*: bool, optional
`-total_current` <*input_current_file*> <*output_total_current_file*>	

	Specifies to create a new current (.ptiavg) file containing the total current for all taps of a net. This parameter can be used if you have a very large current file generated by the Power Calculation engine and want to see only a total current waveform in the SimVision waveform viewer. *input_current_file* is the current file that will be read in (output from the Power Calculation engine). Use the following syntax to create a total current file: `query_power_data -total_current <input_current_file> -out_dir <output_total_current_file>` `-out_dir` is used to specify a user-defined output current filename, and is not a mandatory parameter. If not specified, the output file will be generated automatically with the name of the input file suffixed with `.ptiavg`. *Data_type*: bool, optional

Example

- The following command creates a total current waveform file `VDD` from the input file `dynamic_VDD.ptiavg`:
  ```
  query_power_data -output_tap_name vdd_tap -total_current dynamic_VDD.ptiavg -out_dir VDD
  ```

Related Information

- `report_power`
- power Category Attributes
- "Static Power, IRdrop and EM Analysis" in the *Voltus Stylus Common UI User Guide*
- "Dynamic Power and IRDrop Analysis" in the *Voltus Stylus Common UI User Guide*

read_activity_file

```
read_activity_file activity_file
[-end { time1 time2 ...timen } ]
[-format {vcd | tcf | saif | fsdb | phy | shm}]
[-hierarchy_separator separator ]
[-reset]
[-start { time1 time2 ...timen } ]
[-write_net_freq {true | false}]
[-block block_name ]
[-scope scope_name ]
[-scale_time scalefactor ]
[-start_time_shift value ]
[-rtl {true|false}]
[-name_mapping_rule file]
[-weight value]
[-zero_delay {true|false}]
[-cell cell_name ]
```

Specifies the name and type of activity file to be used as input. This command must be used when specifying activity files for static or dynamic power calculation.

Parameters

`-block block_name`	
	Specifies the name of the FSDB/SAIF/TCF/VCD block to map a sub-block activity file with a top level Verilog file. Provides the ability to su files based upon block instance names. The block argument is added to the FSDB/SAIF/TCF/VCD name before comparing it to the Verilo *Data_type*: string, optional
`-cell cell_name`	

	Specifies the master cell name for the activity file. This parameter allows you to support master-cell-based activity specification. That is, yo the master cell name for the activity file, instead of specifying all the instances that were instantiated from a master cell.
	Example
	Let us consider a design that has a cell named "super_filter" with four block instances "super_filter1", "super_filter2", "super_filter3",and "s
	Instance-Based Use Model:
	`read_activity_file -format VCD -block super_filter1`
	`read_activity_file -format VCD -block super_filter2`
	`read_activity_file -format VCD -block super_filter3`
	`read_activity_file -format VCD -block super_filter4`
	Master-Cell-Based Use Model:
	`read_activity_file -format VCD -cell super_filter`

`-end { time1 time2 ...timen }`	

When you use the vector-driven method of performing dynamic instance-based power-consumption calculation, you can specify the part o which you want power analysis to use for the power calculation. This option specifies the end time of this range and is used in conjunctio option.

Specifies the end time as time-value pairs to report the average power across non-overlapping multiple windows specified in the activity f hierarchy, or other part of the design that you want to analyze. Units are in seconds (`s`), milliseconds (`ms`), microseconds (`us`), nanosecond (`ps`).

Default: The time unit is defined in the first `.lib` file read during design import.

You can specify the `$worst_power_window_end` variable with the `-end` parameter to take the worst case end time from the last vector profile TCF and SAIF.

Note: You can read multiple windows in a single VCD file by specifying multiple pairs of the start time and end time for the non-overlappir specified in the activity file. The use model for reading multiple windows is:

```
read_activity_file -format VCD design.vcd -start {start1 start2 start3} -end {end1 end2 end3}
```

The *start1/2/3* and *end1/2/3* values should be in the ascending order.

Multiple current waveforms representing multiple VCD windows are merged into a single continuous current waveform. To capture the wa the edge of user windows, a buffer time is added between the windows being stitched using the `set_power_analysis_mode -settling_bu` buffer time ensures that the stitched waveform is contiguous. The generated waveform hence has the duration of user windows and the in buffers. The default settling buffer period is 400ps.

Data_type: string, optional

`-format {vcd | tcf | saif | fsdb | phy | shm}`

Specifies the net activity of the design using an activity file in the VCD, TCF, FSDB, SAIF, PHY, or SHM format. The arguments specified parameter are case-sensitive.

- The VCD file format uses the toggle count information as the basis for the power consumption calculation. These files can be in compr using the .gz extension.
- The TCF file format contains data on the switching activity of the nets in the design. The switching activity includes the toggle count inf probability of the net or pin being in the logic 1 state.
  These files can be in compressed (zipped) format using the .gz extension.

  Supports extended TCF which contains transition density for X and Z as shown in the following extended TCF example:

  "out" :"0.0 0" "0.005000 2" "0.0 0"

  The fields are defined as follows:

  Field 1: net or pin name
  Field 2: duty and transition density for transition of 0-1
  Field 3: duty and transition density for transition of X(0-X-1)
  Field 4: duty and transition density for transition of Z(0-Z-1)
- The SAIF file format contains aggregate switching activity information on the nets and/or ports in the design. It does not contain time-ba file format is used for quick average power analysis of the design.
- The FSDB file format specifies the net activity of the design using an activity file in FSDB format. Innovus supports the FSDB version u
  Use Model

  ```
  read_activity_file -format fsdb fsdbfile
  report_power
  ```

  (or)

  ```
  report_vector_profile
  ```

  Example

  ```
  read_activity_file -format fsdb test.fsdb -scope top
  ```
- The PHY (Palladium database format generated in the Palladium environment) format contains the simulation activity data generated simulation. The PHY format can be used in all vector-based power analysis flows (vector profiler, static power analysis and dynamic v analysis).
  Use Model

  ```
  read_activity_file-format phy <path to PHY database directory>
  ```

  Example

```
read_activity_file -format phy ./phy2/trace.phy -scope test
```

Notes:

- The xeDebug software must be present in the `PATH` environment. Add xeDebug in the path using the following command:
  ```
  set path = (<install_dir>/bin $path)
  ```
  The xeDebug software version should be the same as the one using which the PHY database is generated.
- The PHY database generated from the following UXE/VXE version is supported:
  ```
  UXE18.6.0 onwards
  VXE18.6.0 onwards
  ```
- When reading a PHY format activity file, you require a DPA (DPA_ToggleCountFormat) license.
- For information on error messages displayed while reading the PHY database, refer to the `xe.msg` file in the PHY database directory
- Make sure that the PHY database is not locked before invoking the Voltus software.
- Make sure that the xeDebug software is not already running on the machine.
- Ensure that the following directories are present in the top-level PHY database directory: `*.phy`, `PDB`, `cellList`, `QTDB`, `dbFiles`, and
- It is recommended that the write permission is assigned on the entire PHY database. In case the write permission cannot be grante `SWFV_FREEZE_PHY` environment variable to `1`.
- If you want to run parallel Voltus executables on the same top-level PHY directory with multiple `*.phy` databases, it is suggested to the multiple top-level directories, with links to the original PHY database. This will allow local copies of the PHY database to acquir

- The Simulation History Manager (SHM) database format is a record of the data signal changes that occur during design simulation. Th consists of a directory, typically with the "`.shm`" suffix. The directory contains two disk files, namely `*.trn` and `*.dsn`. For more informati format, refer to the *SimVision Analysis Environment* manual available on the Cadence Online Support (COS) website.
  Example
  ```
  read_activity_file -format shm  /../shm/ADDER.trn -scope adder_e/u1 -start 0ns -end 60ns
  ```

Data_type: string, optional

`-hierarchy_separator` *separator*

Specifies the separator character in the hierarchical net names, bus names, and pin names for the block or other part of the design that yo must be the same as the separator character in the design netlist.

Default: slash (/)

Data_type: string, optional

`-name_mapping_rule` *file*

Specifies to enable rule-based mapping to map RTL vectors to gate-level netlist. This parameter helps in resolving name mismatch betwe netlists.

The name mapping rule file format is:

```
Name_Mapping_Rule_file
# RTL        # Gate
<rtlstring> <gatestring>
```

where, each line is a rule. The string in the first column(RTL) is what we want to replace with the string in the second column(Gate). The r added onto the previous rules.

For example, to replace all the square `[]` brackets with under score _ for RTL-to-Gate matching, the rule file will be:

```
#RTL    #GATE

[         _

]         _
```

Note: `-name_mapping_rule` is not supported when the `power_comprehensive_automapping` attribute is set to `true`.

Data_type: string, optional

`-reset`

If `-reset` is specified, all previous activity files specified using the `read_activity_file` command will be reset and ignored.

Default: All the previous activity files specified using the `read_activity_file` command are loaded and used (They are additive).

Data_type: bool, optional

`-rtl {true|false}`

	Specifies to process the vector as an RTL vector and perform state propagation. The propagation behavior is global and the full design is vectors are zero-delay vectors so delay annotation is always done for signals from this vector file. *Default*: `false`
`-scale_time` *scalefactor*	
	Scales the FSDB/SAIF/TCF/VCD duration value by the specified scale factor. For example, an FSDB scale of 2 will double the FSDB du **Note:** The `-start` and `-end` parameters of the `read_activity_file` command are post scaled. *Data_type*: string, optional
`-scope` *scope_name*	

	Specifies the FSDB/SAIF/TCF/VCD scope, that is, the name of the module within the activity file associated with the block or other part of want to analyze.
	The scope argument is removed from the FSDB/SAIF/TCF/VCD name before comparing it to the Verilog file.
	⚠ If the scope value has backslash '\' or square brackets '[]', you must specify the scope within curly brackets {{…..}} to ensure that '\' o processed by the Tcl Interpreter. In the following example, the scope value is specified within curly brackets as it contains '\' and '[]':
	`read_activity_file -scope {{reg_based_buffer.reg_inst\/sv_xpol_trans_q_reg[0][14][7][sv_re][4]}}`
	If '\' and '[]' are not there, curly brackets are not required.
	Example
	The following command specifies the u1 module within the VCD file as the module associated with the part of the design being analyzed, `dut_5buf_full_chip` block instance as the block to map:
	`read_activity_file \`
	`-format VCD \`
	`-scope adder/u1 \`
	`-start {0ps 2000ps} \`
	`-end {1100ps 3100ps} \`
	`-block ../vcd/dut_5buf_full_chip`
	Data_type: string, optional

`-write_net_freq {true | false}`	

	This option should be set when an activity from a TCF or VCD file needs to be used to capture net frequency for signal EM analysis (`chec` addition, you must use the `check_ac_limits -use_db_freq` option to read the toggle rates from the activity file.
	Default: `false`
	Example
	The following example reads a TCF file which is used to capture net frequency for signal EM analysis:
	`read_activity_file -format tcf -write_net_freq true design.tcf`
	`check_ac_limits -use_db_freq -report signal_em.rpt`
	Data_type: enum, optional

`-start {` *time1 time2 ... timen* `}`	

	When you use the vector-driven method of performing dynamic instance-based power-consumption calculation, you can specify the part o which you want power analysis to use for the power calculation. This option specifies the start time of this range and is used in conjunctio Specifies the start time as time-value pairs to report the average power across non-overlapping multiple windows specified in the VCD file hierarchy, or other part of the design that you want to analyze. Units are in seconds (`s`), milliseconds (`ms`), microseconds (`us`), nanosecond (`ps`). *Default*: The time unit is defined in the first `.lib` file read during design import. You can specify the `$worst_power_window_start` variable with the `-start` parameter to take the worst case start time from the last vector p required for TCF and SAIF. **Note**: You can read multiple windows in a single VCD file by specifying multiple pairs of the start time and end time for the non-overlappir specified in the activity file. The use model for reading multiple windows is: `read_activity_file -format VCD design.vcd -start {start1 start2 start3} -end {end1 end2 end3}` The *start1/2/3* and *end1/2/3* values should be in the ascending order. Multiple current waveforms representing multiple VCD windows are merged into a single continuous current waveform. To capture the wa the edge of user windows, a buffer time is added between the windows being stitched using the `set_power_analysis_mode -settling_bu` buffer time ensures that the stitched waveform is contiguous. The generated waveform hence has the duration of user windows and the in buffers. The default settling buffer period is 400ps. *Data_type*: string, optional
`-start_time_shift` *value*	
	Specifies to shift the start time of the specified activity files. When you specify the `-scale_time` parameter with the `-start_time_shift` parameter, it scales the start time shift along with the start/end *Data_type*: string, optional
activity_file	The name of the activity file. *Data_type*: string, optional
`-weight` *value*	

	Specifies the weight number of the TCF file. `1.0` is the default value. The value should be a floating number that is smaller than `1.0`. This p to merge multiple TCF files at different function modes for static power estimation of a design.
	
	In the following example, TCF *file1*, *file2*, and *file3* has the weight numbers 0.3, 0.2, and 0.4, respectively.
	`read_activity_file -format TCF file1 -weight 0.3`
	`read_activity_file -format TCF file2 -weight 0.2`
	`read_activity_file -format TCF file3 -weight 0.4`
	Here, *file1*, *file2*, and *file3* will be merged together to represent the toggle and duty for the nets/pins in the design. The relative weight of 0. 0.4/0.9 would be used for static power calculation. In this example, 0.9 is the sum of weights.
	
	Data_type: string, optional

`-zero_delay {true|false}`	
	Specifies to process the vector as a zero-delay vector and perform delay annotation. *Default* : `false`

Activity Precedence

The following commands can define activity:

```
read_activity_file file
     [-format {vcd | tcf | saif} ]
     [-hierarchy_separator separator ]
     [-start time ]
     [-end time ]
     [-reset]
     [-scope scope_name ]
     [-block block_name ]

set_default_switching_activity
    [-input_activity factor ]
```

```
    [-sequential_activity factor ]
    [-period value ]
    [-duty value ]
    [-global_activity factor ]
    [-hierarchy hierarchy _name]


set_switching_activity
    [-reset]
    [-clock clock_name]
    [-period value ]
    [-activity factor | -transition_density ]
    [-net net_name | -port port_name | -pin pin_name ]
    [-duty value ]
    [-instance]
```

The precedence of activity is as follows:

1. User-Defined: Activities applied to specific nets/pins/ports
2. Activity File: Activity specification through a VCD/FSDB/SAIF/TCF format
3. Clock Gates Output Activity: Activity specification to the output of all clock gates
4. SDC/TWF constants: Set constants to a list of pins or ports for use by the timing engine (timing analysis)
5. Hierarchical Global Activity: Activity specification for a specific hierarchy
6. Global Activity: Activity specification for all instances that are part of the data network.
7. Sequential Element Activity: Activity specification applied to the output of all sequential elements.
8. Primary Input Activity: Activity specification for all primary inputs

If you specify multiple activity data for the same items, the power engine will use the one with the higher precedence.

Examples

- The following command reads the compressed ap_wait_test_pll1_mod.vcd.gz VCD file that contains the toggle count information for the power consumption calculations, specifies the crm_ap module within the VCD file as the module associated with the part of the design being analyzed, specifies the slash character (/) as the hierarchical separator for hierarchical net names, bus names, and pin names for the part of the design being analyzed, and specifies the start (180002ns) and end (189802ns) times from the VCD file for the part of the design being analyzed:

  ```
  read_activity_file -format vcd \
  -scope testbench/top/ap/mcu_platform/crm_ap \
  -hierarchy_separator \
  -start 180002ns \
  -end 189802ns \
  ap_wait_test_pll1_mod.vcd.gz
  ```

- The following command reads the compressed A.vcd.gz VCD file that contains the toggle count information for the power consumption calculations, specifies A1 as the block to map a sub-block VCD file with a block instance at the top-level, and specifies the start (180002ns) and end (189802ns) times from the VCD file for the part of the design being analyzed:

  ```
  read_activity_file -format vcd \
  -block ap/A1 \
  -start 180002ns \
  -end 189802ns \
  A.vcd.gz
  ```

- The next example specifies a VCD activity file called dmac_mac.vcd for the scope top/dma_dut.

  ```
  read_activity_file -format vcd -start 10ns -end 20ns \
  -scope top/dma_dut dmac_mac.vcd
  ```

- The following command specifies multiple pairs of the start time and end time for the multiple windows in the VCD file：

  ```
  read_activity_file \
  -format  VCD \
  -scope  adder/u1 \
  -start  {0ps 2000ps} \
  -end    {1100ps 3100ps} \
  ```

```
-block  ../vcd/dut_5buf_full_chip.vcd
```

- The following command specifies to read the RTL-level VCD activity file:

```
read_activity_file -format VCD -scope scope_top -start 300ns -end 700ns /DesignData/VCD/rtl.vcd -block inst_C1 -rtl true
set_db power_method vector_profile power_vector_profile_mode transient power_worst_window_type full power_grid_libraries
/DesignData/PGV/library_pv.cl
```

Related Information

- set_default_switching_activity
- set_switching_activity
- "Static Power, IRdrop and EM Analysis" in the *Voltus Stylus Common UI User Guide*
- "Dynamic Power and IRDrop Analysis" in the *Voltus Stylus Common UI User Guide*

read_activity_map_file

```
read_activity_map_file
[-help]
[-golden {rtl | gate}]
[-gate_block block_name]
[-reset]
[-rtl_block block_name]
[-rtl_to_gate mapping_file ]
-two_column_mapping_file mapping_file
```

Specifies instance name mapping between RTL netlist and GATE level netlist. Power analysis uses this instance name mapping to annotate the output nets of this instance from the pin-based RTL VCD or TCF. The mapping file used for instance name mapping is generated by Conformal.

You must specify this command before `report_power`.

Parameters

`-gate_block block_name`	
	Specifies the GATE activity block name to map the block-level RTL mapping files to the top-level RTL activity files. The GATE block name is appended before the GATE level entries in the mapping file to match the entries in the top-level RTL activity file. *Data_type*: string, optional
`-golden {rtl | gate}`	
	Specifies whether RTL or Gate is Golden. The one not specified as golden will be considered as Revised. Currently the Conformal MatchPoint file does not specify this. *Default*: `rtl` *Data_type*: string, optional
`-help`	Outputs a brief description that includes the type and default information for each `read_activity_map_file` parameter. For a detailed description of the command and all of its parameters, use the man command `man read_activity_map_file`.
`-reset`	Specifies to reset and ignore all the previous mapping files specified using the `read_activity_map_file` command. *Default* : All the previous mapping files specified using the `read_activity_map_file` command are loaded and used. *Data_type*: bool, optional
`-rtl_to_gate mapping_file`	
	Specifies instance name mapping between RTL netlist and GATE level netlist. RTL level VCD or TCF are obtained from simulation on RTL Verilog netlist. Power analysis uses this instance name mapping to annotate the output nets of this instance from the pin based RTL VCD or TCF. The mapping file used for instance name mapping is generated by Conformal. *Data_type*: string, optional
`-rtl_block block_name`	
	Specifies the RTL activity block name to map the block-level RTL mapping files to the top-level RTL activity files. The RTL block name is appended before the RTL level entries in the mapping file to match the entries in the top-level RTL activity file. *Data_type*: string, optional
`-two_column_mapping_file mapping_file`	

	Specifies a file that contains the instance name mapping between the RTL netlist and GATE level netlist.
	This parameter can be used to specify a mapping file instead of the LEC generated mapping file (`-rtl_to_gate mapping_file`) to map points from RTL to Gate netlist. The format of the two column mapping file is:
	`#Gate name =>  # RTL name`
	`gate_inst_name1  => rtl_inst_name1`
	`...`
	Here, the first column corresponds to the gate name and the second column corresponds to the RTL name.
	Data_type: string, required

Example

- The following command specifies a mapping file rtl_map:
  ```
  read_activity_map_file -rtl_to_gate rtl_map
  ```
- The following command performs instance name mapping between RTL netlist and GATE level netlist, wherein the GATE level netlist is Golden and the RTL netlist is Revised; it specifies the mapping file MapFile.alt
  ```
  read_activity_map_file -rtl_to_gate MapFile.alt -golden gate
  ```
- The following commands appends the RTL block name and GATE block name before the RTL and GATE level entries in the mapping file to match the entries in the top-level RTL activity file:
  ```
  read_activity_map_file -rtl_to_gate LEC.top \
  -rtl_block  "core/block_top" \
  -gate_block "core/block_top"
  ```
- The following commands specify to ignore the previous mapping file (`old.map`), and use the modified mapping file (`updated.map`):
  ```
  read_activity_map_file -rtl_to_gate old.map
  read_activity_map_file -reset
  read_activity_map_file -rtl_to_gate updated.map
  ```

An example of the mapping file output from Conformal is described as follows:

```
Mapped points: SYSTEM class
1-th mapped points:
 (G) + 1 PI /x_ick
 (R) + 595 PI /x_ick
2-th mapped points:
 (G) + 2 PI /x_jreset_cp_p
 (R) + 594 PI /x_jreset_cp_p
3-th mapped points:
 (G) + 3 PI /x_mreset_cp_p
 (R) + 593 PI /x_mreset_cp_p
........
60-th mapped points:
 (G) + 4582 DFF /cpexec0/cpddecls0/gi_opls/q_reg_reg[35]
 (R) + 3856 DFF /cpexec0/cpddecls0/gi_opls/q_reg_reg_35_

61-th mapped points:
 (G) + 4583 DFF /cpexec0/cpddecls0/gi_opls/q_reg_reg[34]
 (R) + 3855 DFF /cpexec0/cpddecls0/gi_opls/q_reg_reg_34_

62-th mapped points:
 (G) + 4633 DFF /cpexec0/cpddecex0/gi_opex/q_reg_reg[21]
 (R) + 3805 DFF /cpexec0/cpddecex0/gi_opex/q_reg_reg_21_

63-th mapped points:
 (G) + 4634 DFF /cpexec0/cpddecex0/gi_opex/q_reg_reg[20]
 (R) + 3804 DFF /cpexec0/cpddecex0/gi_opex/q_reg_reg_20_

64-th mapped points:
 (G) + 4635 DFF /cpexec0/cpddecex0/gi_opex/q_reg_reg[19]
 (R) + 3803 DFF /cpexec0/cpddecex0/gi_opex/q_reg_reg_19_

65-th mapped points:
 (G) + 4636 DFF /cpexec0/cpddecex0/gi_opex/q_reg_reg[18]
 (R) + 3802 DFF /cpexec0/cpddecex0/gi_opex/q_reg_reg_18_
```

where, `G` refers to the Golden netlist name and `R` refers to the Revised netlist name.

By default, the tool assumes that the Golden netlist name refers to the RTL net name and Revised netlist refers to the GATE level net name.

Related Information

- `report_power`
- `read_activity_file`
- "RTL Activity File Flow in Static Power Calculation" in the *Voltus Stylus Common UI User Guide*
- "Dynamic Power and IRDrop Analysis" in the *Voltus Stylus Common UI User Guide*

read_power_db

```
read_power_db
[-help]
-in_file {file1 file2 file3 ...}
[-hierarchy {prefix1 prefix2 prefix3 ...}]
[-hierarchy_separator "/"]
```

Restores any `power.db` file in the Voltus/Innovus session. This command can be used after restoration of a design to view the static power results or incremental report generation without having to do power analysis again.

Note: You will be able to restore the power database that you created previously, only if you created the power database using the `power_write_db` attribute. Also, the command loads power databases created in the 9.1 release only, and not in the older releases.

- The command can map a hierarchical instance to a flat netlist.
- It can restore multiple power dbs for blocks and top level, that are generated separately.
- When the information for a net or an instance is present in multiple power dbs, the last one will overwrite.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `read_power_db` parameter. For a detailed description of the command and all of its parameters, use the man command `man read_power_db`.
`-in_file {file1 file2 file3 ...}`	
	Specifies the name of the power database file. *Data_type*: string, required
`-hierarchy {prefix1 prefix2 prefix3 ...}`	
	Specifies to map each instance in the file with a prefixN/instance (for example, prefixA/inst) in the flat netlist, where '/' is the hierarchy separator. This is an optional parameter. *Default*: `all` *Data_type*: string, optional
`-hierarchy_separator "/"`	
	Specifies the separator character in the hierarchical net names, bus names, and pin names for the block or other part of the design that you want to analyze. It must be the same as the separator character in the design netlist. This is an optional parameter. *Default*: slash (`/`) *Data_type*: string, optional

Use Model

```
read_db / read_design
read_power_db -in_file power.db
report_power # for text based power reports
```

Example

For each instance in file 'file2' a prefix "A" is added to instance name such as instC, A/instC. No prefix is added to `file1' since prefix1 is none for file1.

```
read_power_db -in_file {file1 file2} -hierarchy {none A}
```

Related Information

- `report_power`
- power Category Attributes
- "Static Power, IRdrop and EM Analysis" in the *Voltus Stylus Common UI User Guide*

report_inst_power

```
report_inst_power
[-help] instance [-out_file filename]
```

Generates a text based report for the specified instance. This command enables you to determine how the different components of static power are calculated. You can use the report to debug static power numbers after performing power analysis (`report_power`).

This command is available through both the static and dynamic power engines.

Note: For dynamic power engine, the `report_inst_power` command does not generate text reports incrementally. If the dynamic run has been performed and you need the debug information for a static power number, you should re-run power analysis.

Parameters

`-out_file` *filename*	Writes out the static power analysis report into the file that you specify. *Data_type*: string, optional
`-help`	Outputs the command usage and a brief description about the command parameters.
instance	Specifies the instances to include in the instance power report. *Data_type*: string, required

Example

- The following command specifies an instance power report file `debug.rep` for the instance `top/i_1`:
  ```
  report_inst_power -out_file debug.rep top/i_1
  ```

Related Information

- `report_power`
- power Category Attributes
- "Static Power, IRdrop and EM Analysis" in the *Voltus Stylus Common UI User Guide*
  - "Debugging Instance Power"

report_power

```
report_power
[-cap]
[-cell {cell_list}]
[-cell_type {all | {macro io combinational sequential clock_combinational clock_sequential}}]
[-clock_domain names]
[-clock_network {all | {clock_list}}]
[-exclude_cells {cell_list}]
[-exclude_cells_file filename ]
[-exclude_insts {instance_list}]
[-exclude_instances_file filename ]
[-include_sequential_in_clock]
[-insts {instance_list}]
[-hierarchy {all | hierarchy_level}]
[-leakage]
[-net [-nworst number_of_nets]]
[-no_wrap]
[-out_file filename]
[-pg_net {all | pg_net_name_list}]
[-power_domain {all | {power_domain_list}}]
[-sort {internal | switching | leakage | total}]
[-threshold value]
[-view view_name]
[-threshold_voltage_group { all | group_name}]
[-clock_gating_efficiency]
[-register_gating_efficiency]
[-cluster_gating_efficiency]
[-thermal_leakage_temperature {temp1 temp2 temp3 …}]
[-thermal_power_map_file file_name]
[-thermal_power_map_tiles {xint yint}]
[-thermal_power_map_format {simple | stack }]
[-pg_pin]
[-thermal_conductivity_inputs file_name]
[-toggle_rate]
[-combinational_sequential_power_distribution]
[-format { simple | detailed }]
[-out_dir directory]
[-report_prefix prefix]
[-hierarchical_instances {inst1 inst2 inst3 …}]
[-group_type user_defined_group_name]
[-distribute]
[-print_memory_power]
[-cap_unit unit]
[-cell_list_file filename]
[-float_precision value]
[-inst_list_file filename]
[-power_db_directory directory]
[-power_unit unit ]
[-time_unit unit]
[-power_density_tiles value]
[-power_density_tiles_row_column {value1 value2}]
[-power_density_tiles_size {value1 value2}]
[-compress compression_ratio]
[-time_based_report]
[-thermal_power_map_header_include_file filename]
[-thermal_leakage_temperature_scale_table_file filename]
[-thermal_leakage_temperature_scale_table {$temp1 $scalefactor1 $temp2 $scalefactor2 $temp3 $scalefactor3 .....}]
[-thermal_material_file filename]
[-thermal_power_map_si_unit {true | false}]
[-thermal_power_map_power_table_tiles {Xint Yint}]
[-thermal_power_map_density_table_tiles {Xint Yint}]
[-thermal_power_map_format_version {new|default}]
[-thermal_power_map_bottom_layer layer_name]
[-thermal_power_map_top_layer layer_name]
[-use_geometry_cell_size {true | false}]
```

Reports global as well as specific instance power, clock network power, clock domain power, and net switching power. It also reports the power for power domains and specific power nets. Units are reported in milliwatts.

This command starts the execution of power analysis (both static and dynamic). The `report_power` command has a character limit of 1015.

> The parameters of the `report_power` command, except `-cap`, are not supported in the dynamic only mode (specified using the `power_disable_static` attribute). These parameters are only supported in the static mode. However, `-clock_gating_efficiency`, `-register_gating_efficiency`, and `-power_domain` parameters are not supported even if the static power mode is enabled.

The `report_power` command allows generation of multiple power reports with a single `report_power` command. This allows you to generate all the static reports during dynamic analysis, and removes the need to run incremental `report_power` in static power analysis using a Verilog netlist. By default, a detailed power summary report, and a short summary will be printed at the beginning of all reports. To enable this flow, you need to specify the following parameters:

- `-out_dir`
- `-report_prefix`

The following `report_power` command parameters are supported by the multi-report flow, both in static and dynamic power analysis:

- `-cell`
- `-clock_gating_efficiency`
- `-register_gating_efficiency`
- `-cluster_gating_efficiency`
- `-net`
- `-insts`
- `-pg_net`
- `-clock_network`
- `-hierarchy`

The following parameters are not supported in the multi-report flow:

- `-cell_type`
- `-power_domain`
- `-clock_domain`
- `-clock_gating_efficiency` and `-clock_network` in the same run

> In the static power analysis flow, you can run the `report_power` command after the `read_power_db` command without loading the design. The following command is an example of the use model:
>
> ```
> read_power_db -in_file power.db
> report_power -out_dir same -report_prefix report -clock_network {all} -instances {*}
> ```
>
> The following parameters of the `report_power` command are supported without design loading:
>
> - `-cell`
> - `-cell_type`
> - `-insts`
> - `-clock_network`
> - `-clock_domain`
> - `-pg_net`
> - `-pg_net -format detailed`
> - `-hierarchy`
> - `-hierarchical_instances`

Parameters

`-cap`	Dumps out the list of net, pin, and total capacitances in three different columns. *Data_type*: bool, optional
`-cap_unit` *unit*	
	Sets the capacitance unit to be used for power report generation. The default unit is pF (picoFarads). *Data_type*: string, optional
`-cell` {*cell_list*}	
	Specifies the cells to include in the power report. Accepts wildcards (*). *Data_type*: string, optional
`-cell_list_file` *filename*	
	Specifies a file containing a list of cells that are to be included in the power reports. *Data_type*: string, optional
`-cell_type {all | {macro io combinational sequential clock_combinational clock_sequential}}`	
	Specifies the cell type to include in the power report. *Default*: `all` When you specify the cell types, `-cell_type combinational` or `sequential`, the power report now does not show the clock combinational or cl *Data_type*: string, optional
`-clock_domain` *names*	
	Specify clock domains in power reports. **Usage** `report_power -clock_domain` *name* `-out_file` *filename*
`-clock_gating_efficiency`	

	Generates a report that includes the Clock Gating Efficiency (CGE) for all clock gating instances as well as different hierarchies in the design different hierarchies is also reported.
	Clock Gating Efficiency is calculated as:
	`CGE = (toggles at clock gate output/toggles at clock gate input)`
	`Average CGE = (avg of all CGEs above)`
	When you generate a CGE report using the command `report_power -clock_gating_efficiency -out_file` *cge.rpt*, the following informatic
	• **Clock-gating Efficiency Report** - for each clock domain, it includes the toggle rate, number of registers, number of clock gates, average c registers, average toggle savings at registers, and average toggle savings histogram.
	• **Hierarchical View of Average Toggle Savings** - number of clock gates and average toggle savings for each hierarchical module in the c
	Data_type: bool, optional

`-clock_network` {`all` | {*clock_list*}}	
	Reports the power consumption of the clock network, including the power for generated clocks. Use `clock_list` to report the power consumed by specific clocks. *Default*: `all` *Data_type*: string, optional
`-cluster_gating_efficiency`	

	Specifies to generate the cluster efficiency report in the RGE report. This report gives the CGE/RGE metrics for the registers at the fanout of e instance in the design. *Data_type*: bool, optional
`-combinational_sequential_power_distribution`	
	Specifies to generate a report that gives the ratio of combinational versus sequential power for all modules of a given hierarchy. You must spe with the `-insts` parameter. *Data_type*: bool, optional
`-compress` *compression_ratio*	
	Specifies to compress the power analysis report files in the GNU Zip format. The compressed files will have the suffix `.gz`. *compression_ratio* is from 0-9. `0` indicates that the report files are not to be compressed. The compression levels vary from `1` to `9`. `1` provides f compression speed but at a lower ratio, so the file size will be larger. `9` provides the highest compression ratio but at a lower speed. *Default*: 0 *Data_type*: int, optional
`-distribute`	Enables standalone distributed power analysis. This parameter allows you to perform dynamic power analysis in the multi-CPU mode withou *Data_type*: bool, optional
`-exclude_cells {`*cell_list*`}`	
	Specifies a list of cells that should be excluded from power analysis. *Data_type*: string, optional
`-exclude_cells_file` *filename*	
	Specifies a file containing a list of cells (one per line) which should be excluded from power analysis. *Data_type*: string, optional
`-exclude_insts {`*instance_list*`}`	
	Specifies a list of instances that should be excluded from power analysis. *Data_type*: string, optional
`-exclude_instances_file` *filename*	
	Specifies a file containing a list of instances (one per line) which should be excluded from power analysis. *Data_type*: string, optional
`-float_precision` *value*	
	Sets the precision value to be used for power report generation. The default value is 4. *Data_type*: float, optional
`-format { simple | detailed }`	
	Specifies the type of report summary required for static power reporting. The possible arguments of this parameter are: • `simple` - reports the amount of power consumed by all power nets in a condensed format, breaking down power based on the switching/in components. • `detailed` - reports the amount of power consumed by all power nets in a detailed format. The detailed format includes additional informatic capacitance (`Cpin Sum`), output net capacitance (`Cnet Sum`), Total Load Capacitance (`Cpin Sum` + `Cnet Sum` = `Total Load_Cap`), maximum f and output pins (`Max_Clk_Freq`), maximum activity factor at the input and output pins (`Max_Activity`), cell name of the instance (`Cell Name`) (`C/D`), and library used in power analysis for the specific cell (`Lib`). The libraries are numbered in the header section. The name of the generated report file is `report.*.detailed.rpt`. Currently, the `-format` detailed report is only available for the `-pg_net` report *Data_type*: string, optional
`-include_sequential_in_clock`	

	Includes the leaf flip-flop power in the clock network power report. *Default*: The leaf flip-flop power will not be included in the clock network power report. *Data_type*: bool, optional
`-hierarchical_instances {`*`inst1 inst2 inst3 …`*`}`	
	Specifies to report all the power content for the specified hierarchical instances/ blocks. It reports the `Total Power` (a breakup of power compo breakup of power by cell type) of the specified block at each hierarchy. To generate a complete hierarchical instance power report, it is recommended to specify `-hierarchical_instances` with `-report_prefix`. The specified output directory. If you specify `-outfile` instead of `-report_prefix`, an instance file is output without hierarchy-level details. `-report_prefix` generates the following 3 reports: • `<`*`prefix`*`>.rpt` - reports power details for all instances in the design. • `<`*`prefix`*`>.hierpwr.rpt` - reports hierarchical power, that is, power summary, internal, switching, leakage, and total power for all hierarchica • `<`*`prefix`*`>.hierinstpwr.rpt` - reports power details for the requested block instances. **Note**: `-hierarchical_instances` must be specified with `-hierarchy` to include hierarchy-level power report. When `-hierarchical_instances hierarchy`, only top-level summary of the power consumption is written. *Data_type*: string, optional
`-hierarchy {all |` *`hierarchy_level`*`}`	
	Specifies the hierarchy level that is included in the power report. You can specify the *`hierarchy_level`* as a number that corresponds to a spe the design. A *`hierarchy_level`* of 0 specifies the top level. *Default*: `all` *Data_type*: string, optional
`-insts {`*`instance_list`*`}`	
	Specifies the instances to include in the power report. The power report provides the amount of power consumed by a specified list of hierarc instances. *`instance_list`* accepts wildcards (*) for leaf-level instances only. An instance with 0 power will not be listed in the power report. *Data_type*: string, optional
`-inst_list_file` *`filename`*	
	Specifies a file containing a list of instances that are to be included in the power reports. *Data_type*: string, optional
`-leakage`	Reports only leakage power of the design. *Data_type*: bool, optional
`-net`	Reports the net switching power, as well as the load capacitance, toggle rate, and switching values for each net in the design. *Data_type*: bool, optional

`-no_wrap`	Displays the `report_power -cell_type all` output on a single line in the report output file.
	Example
	`Instance Toggle Internal Switching Leakage Total Percentage Cell`
	`Power Power Power Power (%) Name`
	`----------`
	`dma_dut/dmamaster_dut/internal_memory/RAM4096x32/ram0 7.102e+08 5.58 0.002307 0.02424 5.607 18.19`
	`TSDN65ethernet_mac_1/rx_fifo_module_to_dma/data/dual_port_ram_4_tx_fifo/ram2P1024x32/ram`
	Data_type: bool, optional

`-nworst` *`number_of_nets`*	

	Reports the number of nets with the highest net switching power. **Note:** You must use the `-nworst` parameter in conjunction with the `-net` parameter. *Data_type*: string, optional
`-out_file` *filename*	
	Specifies the name of the report file in which the top-level summary of the power consumption information is written. You can include the dire You can convert a power report file to the gzip file format by specifying `-out_file` *filename.gz*. **Note:** If you do not specify this parameter, the software directs the power report to the command line, as well as the log file, and no power rep separate file. *Data_type*: string, optional
`-out_dir` *directory*	
	Specifies the name of the power analysis output directory that saves the power reports. The default is the current working directory. *Data_type*: string, optional
`-pg_net` {`all` | *pg_net_name_list*}	
	When set to `all`, reports the amount of power consumed by all power nets. If you specify a list, reports the power consumed by each instance that is connected to the specified list of power nets. **Note:** When using this parameter, the static power calculation engine always performs propagation and power calculation. *Default*: `all` *Data_type*: string, optional
`-pg_pin`	
	Reports power consumed by all PG pins in a design. *Data_type*: bool, optional
`-power_db_directory` *directory*	
	Specifies the location of the power database that the Power Calculation engine uses for generating standalone power reports. *Data_type*: string, optional
`-power_domain` {`all` | {*power_domain_list*}}	
	Specifies all power domains or lists power domains to be included in the report. It prints a summary of power break down for each domain be each instance in the domain. *Data_type*: string, optional
`-power_density_tiles` *value*	
	Specifies the number of tiles used for power density calculation. *Default*: `100` *Data_type*: string, optional
`-power_density_tiles_row_column` {*value1 value2*}	

Defines the number of tiles by rows * columns. *value1* is the number of tile rows and *value2* is the number of tile columns.

The following example specifies to perform power density aware vector profiling using 2 tile rows and 2 column rows:

```
set_db power_method vector_profile power_vector_profile_mode power_density power_worst_step_size 10ns
report_power  -power_density_tiles_row_column  {2 2} -time_based_report -out_file tiling1.rpt
```

The following is the vector profile report for this command:

```
Tile 0_0_0 5.06 5.33 14.95 14.555
Tile 1_0_1 14.95 5.33 24.84 14.555
Tile 2_1_0 5.06 14.555 14.95 23.78
Tile 3_1_1 14.95 14.555 24.84 23.78
...
...
```

Data_type: string, optional

`-power_density_tiles_size {value1 value2}`

Defines the absolute tile size. *value1* is the horizontal width of a tile (in micron) and *value2* is the vertical height of a tile (in micron).

Data_type: string, optional

`-power_unit unit`

Sets the power unit to be used for power report generation. The default unit is mW (miliAmps).

Data_type: string, optional

`-print_memory_power`

Specifies to report power values for the memory and macro cells under separate groups (`Memory` and `Macro`).

The following is a snapshot of the power report showing separate groups for memory and macro cells:

Group	Internal Power	Switching Power	Leakage Power	Total Power	Percentage (%)
Sequential	0.4031	0.1283	0.006583	0.538	18.38
Memory	0.1624	0.004345	0.011	0.1778	6.074
IO	0	0	0	0	0
Physical-Only	0	0	2.91e-06	2.91e-06	9.945e-05
Combinational	0.8711	1.05	0.01579	1.937	66.2
Clock (Combinational)	0.002302	0.01018	7.654e-05	0.01255	0.429
Clock (Sequential)	0.01177	0.008387	0.000347	0.0205	0.7005
Macro (non-mem)	0.2317	0.007259	0.001234	0.2402	8.208
Total	1.682	1.209	0.03503	2.926	100

Data_type: bool, optional

`-group_type user_defined_group_name`

Specifies to report power (internal/switching/leakage/total) for a user-defined collection of cells that are stated under a power group in the pov
You can use the following Tempus commands to create power groups:

- `define_property` - define a property.
- `set_property` - assign the defined property (group type) and its value (power group) to a list of cells.

The use model of the `-group_type` parameter is:
`report_power -group_type user_defined_group_type`

Here, *`user_defined_group_type`* is the user-defined power group type applied to a list of cells in a design. When the `-group_type` parameter i
command reports power for the power groups associated with the specified group type in the power summary report.

Example:

The following command defines the `poly_bias` property (group type) on library cells:
`define_property poly_bias -obj_type lib_cell -type string`

The following commands apply the property `poly_bias` with the group name `P0` and `P10` to the collection of cells:
`set_property [get_lib_cells -of_objects [get_cells -hierarchical * -filter {ref_name=~*PB0*}]] poly_bias  P0`
`set_property [get_lib_cells -of_objects [get_cells -hierarchical * -filter {ref_name=~*PB10*}]] poly_bias  P10`

The following command reports power for the user-defined power groups that are associated with the property `poly_bias`:
`report_power -out_file basic_group_type.rpt -group_type poly_bias`

The following is a snippet of the power report that shows the user-defined power groups `P0` and `P10` that are associated with the property `poly`
the existing groups (sequential, macro, IO, and so on):

```
User Group                      Internal   Switching     Leakage       Total  Percentage
                                Power      Power         Power         Power  (%)
-------------------------------------------------------------------------------------
P0                               0.01498   0.0002017     0.01765     0.03283       15.63%
P10                               0.1268     0.04026     0.01011      0.1772       84.37%
-------------------------------------------------------------------------------------
Total                            0.1418     0.04047     0.02776        0.21         100%
-------------------------------------------------------------------------------------
```

```
Group                           Internal   Switching     Leakage       Total  Percentage
                                Power      Power         Power         Power  (%)
-------------------------------------------------------------------------------------
Sequential                       0.02958   0.0003768     0.01458     0.04453       16.62
Macro                                  0           0           0           0           0
IO                                     0           0           0           0           0
Combinational                   0.008217    0.002633     0.01025      0.0211       7.877
Clock (Combinational)             0.1239     0.04421    0.007419      0.1755       65.51
Clock (Sequential)               0.01773    0.008783   0.0002459     0.02676       9.988
-------------------------------------------------------------------------------------
Total                            0.1794       0.056     0.03249      0.2679         100
-------------------------------------------------------------------------------------
```

Data_type: string, optional

`-register_gating_efficiency`

Generates a report that includes the Register Gating Efficiency (RGE) and Data Aware Gating Efficiency (DAGE) for all sequential cell instan
hierarchies in the design. Average RGE and DAGE for different hierarchies is reported. It also displays a section on ICG cluster information, t
about registers at the fanout of each ICG instance along with their RGE metrics.

Register Gating Efficiency is calculated as:

```
RGE = 1 - (toggles at register clock pin/root clock toggles)
```

When you generate an RGE report using the command `report_power -register_gating_efficiency -out_file rge.rpt`, the following infor
displayed:

- **Register-gating Efficiency Report** - for each clock domain, it includes the toggle rate, number of registers, number of clock gates, averag
registers, average toggle savings at registers, and average toggle savings histogram.
- **Register Gating Opportunities Report** - this report is sorted based on the Q/CLK toggle ratio. When a register's Q/CLK ratio is greater tha
every other clock cycle. However, when it is less than 25%, there is a gating opportunity to reduce power.

Data Aware Gating Efficiency is calculated as:

```
DAGE = 1 - ((toggles at clock pin - 2*toggles at Q pin)/(root clock toggles))
```

Data_type: bool, optional

`-report_prefix prefix`

Specifies a prefix for the report in which the top-level summary of the power consumption information is written. All the new report names will the detailed summary will be *prefix*.rpt.

Data_type: string, optional

-sort {internal | switching | leakage | total}

Sorts the instance-based power consumption data by internal, switching, leakage, or total power.

Default: total

Data_type: string, optional

-thermal_conductivity_inputs *file_name*

Specifies a file containing the thermal conductivity value for layers. You can use this parameter to include the thermal conductivity value of th the power map file:

- Silicon
- Metal
- Dielectric
- Micro Bump (optional for 3DIC design)
- Fill material in micro bump layer (optional for 3DIC design)

The thermal conductivity file format is:
Section 1 (layer type based thermal conductivity value)

```
#default
$layer_type1 $thermal_conductivity_parameter1
$layer_type2 $thermal_conductivity_parameter2
...
```

Section 2 (layer name based thermal conductivity value)

```
#perlayer
$layer1_name $thermal_conductivity_parameter1 $dielectric_thermal_conductivity_1
$layer2_name $thermal_conductivity_parameter2 $dielectric_thermal_conductivity_2
...
```

Here,

- You can either specify both Section 1 (layer type based) and Section 2 (layer name based), or specify one of the two sections. If both secti layer name based section takes precedence over layer type based section.
- The layer name based section has thermal conductivity value and its dielectric layer's thermal conductivity value for each metal/via layer.
- If section 2 has no parameters defined, the software will use the default parameters defined in the section 1.

Example:
###default

```
Metal      430
Silicon    250
Dielectric 0.6
```

###Perlayer

```
Metal1    430    0.4
Via1      410    0.5
Metal2    390    0.5
Via2      410    0.5
...
```

For information on the thermal conductivity input file and power map file, refer to the "Static Power, IRdrop and EM Analysis" chapter in *Voltu*

Data_type: string, optional

-thermal_leakage_temperature {*temp1 temp2 temp3 …*}

Defines the temperature to calculate leakage for the power map file.

Data_type: string, optional

-thermal_leakage_temperature_scale_table {*$temp1 $scalefactor1 $temp2 $scalefactor2 $temp3 $scalefactor3 .....*}

Specifies the global scale factor for a temperature value to enable scaling of the leakage power values coming from the input dotlib file. This p to scale the power values only at certain specific temperatures.

This parameter allows you to generate a multi-temperature leakage power map table from a single input dotlib file in the power map generatio `thermal_leakage_temperature_scale_table` and `-thermal_leakage_temperature_scale_table_file` are mutually exclusive.

Data_type: string, optional

`-thermal_leakage_temperature_scale_table_file` *filename*

Specifies a file containing the user-defined temperature values and the corresponding scale factors for each library, cell, or instance. The form

```
#Type Name temp1 temp2 temp3 temp4
Lib    library_name   scalefactor@temp1   scalefactor@temp2  scalefactor@temp3   scalefactor@temp4
Cell   cell_name      scalefactor@temp1   scalefactor@temp2  scalefactor@temp3   scalefactor@temp4
inst   instance_name  scalefactor@temp1   scalefactor@temp2  scalefactor@temp3   scalefactor@temp4
```

This parameter allows you to generate a multi-temperature leakage power map table from a single input dotlib file in the power map generatio `thermal_leakage_temperature_scale_table` and `-thermal_leakage_temperature_scale_table_file` are mutually exclusive.

File Format Example for the Type Lib:

```
#Type Name          25 50 125 150
Lib   fast          1 2 10 100
Lib   bufao         1 2 10 100
Lib   header        1 2 10 100
Lib   pso_ring_wc   1 2 10 100
```

Data_type: string, optional

`-thermal_material_file` *filename*

Specifies the name of a file containing the user-specified substrate layer thickness. The layer thickness value for the power map generation fl user-specified value or derived from the technology file. If both the values are available, the user-specified value takes precedence.

Data_type: string, optional

`-thermal_power_map_bottom_layer` *layer_name*

Writes zero metal density tables for layers lower than the user-specified layer in the power map file. This parameter prevents duplication of m when merging the top and block thermal power map files.

Data_type: string, optional

`-thermal_power_map_density_table_tiles {`*Xint Yint*`}`

Specifies the number of tiles in the X and Y directions for the metal density table.

Data_type: string, optional

`-thermal_power_map_file` *file_name*

Generates the power map file with the specified name in the output directory.

Data_type: string, optional

`-thermal_power_map_format {simple | stack }`

Generates a tile-based power map file in the simple or stack format to perform accurate thermal analysis. You must use the stack format to su package for thermal analysis.

Data_type: string, optional

`-thermal_power_map_format_version {new|default}`

	Determines whether to generate the default or the new power map file format. When set to `new`, the power map file format with non-uniform tile power (`-thermal_power_map_power_table_tiles`) and metal density tables (`-thermal_power_map_density_table_tiles`) can be generated. The following are the features of the new report format: • version number added to the header "Version : 2020.0" • nx/ny under the "DIE_AREA" section has been removed • XCOR/YCOR for each layer has been removed • nx/ny has been added for the POWER and METAL_DENSITY sections *Data_type*: string, optional
`-thermal_power_map_header_include_file` *filename*	
	Specifies a file that contains comments or additional information to be included in the header section of the power map file. *Data_type*: string, optional
`-thermal_power_map_power_table_tiles {`*Xint Yint*`}`	
	Specifies the number of tiles in the X and Y directions for the power table. *Data_type*: string, optional
`-thermal_power_map_si_unit {true | false}`	
	Specifies to change chip thermal power map units to SI in the thermal power map file. This parameter was introduced to support the thermal p used by foundry. When this parameter is set to `true`, the values for all the power map parameters (length, power, and thermal conductivity) w units. • **Length Unit:** default is `um` and SI is `m` • **Power Unit**: default is `mW` and SI is `W` • **Metal/Silicon Thermal Conductivity Unit**: default is `mW/(um * C)` and SI is `W/(m * C)` *Default* : `true` *Data_type*: string, optional
`-thermal_power_map_tiles {`*xint yint*`}`	
	Defines the tile number in the X and Y direction. By default 10*10 will be used. *Data_type*: string, optional
`-thermal_power_map_top_layer` *layer_name*	
	Writes zero metal density tables for layers higher than the user-specified layer in the power map file. This parameter prevents duplication of m when merging the top and block thermal power map files. *Data_type*: string, optional
`-threshold` *value*	
	Filters out the instances whose total power dissipation is more than the specified value. Units are in milliwatts. *Default*: `0` *Data_type*: string, optional
`-threshold_voltage_group { all |` *group_name*`}`	
	Reports leakage by voltage threshold group in the power report. This parameter supports the argument `all` (default value) or any other string, *Default*: `all` *Data_type*: string, optional
`-time_based_report`	
	Writes out a detailed VCD profiling report into the file that you specify. *Data_type*: bool, optional

`-time_unit` *unit*	
	Sets the time unit to be used for power report generation. The default unit is ns (nanoseconds). *Data_type*: string, optional
`-toggle_rate`	
	Specifies to dump out instance-based activity to the outfile. The toggle rate for each instance is based on the clock frequency to which it belo are reaching the instance, then the fastest clock is used as a reference. *Data_type*: bool, optional
`-use_geometry_cell_size {true | false}`	
	Specifies whether to use the actual geometry or placement bounding box for macro PGV cell size during generation of tile-based power map used when a macro's power is distributed across multiple tiles. When this parameter is set to `true`, the geometry-based bounding box will be generation and power will be distributed evenly across tiles. *Data_type*: bool, optional
`-view` *view_name*	
	Specifies the active timing view that was created using the multi-mode multi-corner (MMMC) `set_analysis_view -setup|-hold` command. Po performed for the specified view. The command prints the `MMMC` view name in the output file even if you do not specify the `-view` parameter. If `report_power -view` *view_name* is set, it will override the view set in the `power_view` attribute. However, if you specify the `set_analysis_vie leakage` parameters, then the view set using these `set_analysis_view` command parameters has higher priority over the `report_power -view` *Default*: Uses current view in Innovus. *Data_type*: string, optional

Examples

- The following command reports the amount of power consumed by the clock nets clk and test_clk (including leaf flip-flop power) and writes the output information to a power report file named clk_pwr.rep:
  ```
  report_power -clock_network{clk test_clk} \
  -include_sequential_in_clock -out_file clk_pwr.rep
  ```
- The following command reports the amount of power consumed down to hierarchy level 3 from the top and writes the output information to a power report file named hier.rep:
  ```
  report_power -hierarchy 3 -out_file hier.rep
  ```
- The following command compresses the power report file pg_net.rep with gzip:
  ```
  report_power -out_file pg_net.rep.gz
  ```
- The following command reports the amount of power consumed by leaf-level instances that match ECO* and writes the output information to a power report file named inst.rep:
  ```
  report_power -insts ECO* -out_file inst.rep
  ```
- The following command reports the amount of power consumed by all cells that match buf*6 and writes the output information to a power report file named cell.rep:
  ```
  report_power -cell buf*6 -out_file cell.rep
  ```
- The following command reports the amount of power consumed by all I/O and sequential cells and writes the output information to a power report file named cell_type.rep:
  ```
  report_power -cell_type {io sequential} -out_file cell_type.rep
  ```
- The following command sorts the instance-based power consumption data by leakage power and writes the output information to a power report file named sort.rep:
  ```
  report_power -sort leakage -out_file sort.rep
  ```
- The following command reports the total power of all instances whose total power dissipation is less than 1 milliwatt and writes the output information to a power report file named threshold.rep:
  ```
  report_power -threshold 1 -out_file threshold.rep
  ```
- The following command reports the net switching power, load capacitance, toggle rate, and switching values for each net in the design, reports the top 100 nets with the highest net switching power, and writes the output information to a power report file named net.rep:
  ```
  report_power -net -nworst 100 -out_file net.rep
  ```

- The following command reports the amount of power consumed by each power domain and writes the output information to a power report file named pd_all.rep:
  ```
  report_power -power_domain all -out_file pd_all.rep
  ```
- The following command reports the amount of power consumed by each power domain PD2 and writes the output information to a power report file named pd.rep:
  ```
  report_power -power_domain PD2 -out_file pd.rep
  ```
- The following command reports the amount of power consumed by view view_max-hp-max-lp1 and writes the output information to a power report file named view.rep:
  ```
  report_power -view view_max-hp-max-lp1 -out_file view.rep
  ```
- The following command reports the amount of power consumed by each instance connected to power nets vdd and vdd1 and writes the output information to a power report file named pg_net.rep:
  ```
  report_power -pg_net {vdd vddl} -out_file pg_net.rep
  ```
- The following command reports the leakage for the threshold voltage group HVT:
  ```
  report_power -leakage -threshold_voltage_group { HVT }
  ```
- The following command writes out a detailed vector profiling report for the combinational and macro cell types in the `vectorprofile.rpt` file:
  ```
  report_power -time_based_report -out_file vectorprofile.rpt -cell_type {combinational macro}
  ```
- The following command is an example of generating multiple reports at the same time:
  ```
  report_power -out_dir static_power -report_prefix design -net {name1} -insts {inst_name}
  ```
  The following reports will be generated:
  ```
  static_power/design.rpt
  static_power/design.netpwr.rpt
  static_power/design.instpwr.rpt
  ```
- The following command generates a power map file in the simple format for the specified tile:
  ```
  report_power -out_file powermeter.txt -thermal_power_map_file powermap.txt -thermal_power_map_tiles { 4 6 } -thermal_power_map_format simple
  ```
- The following command specifies to report the power content for the `inst1` hierarchical instance in the `report.hierinstpwr.rpt` report:
  ```
  report_power -output static -report_prefix report -clock_network {all} -instances * -no_wrap -hierarchy {all} -hierarchical_instances inst1 -cell_type {all}
  ```
- Example of the output of the power report.
  ```
  *------------------------------------------------------------------------
  *        -  - Version  64-bit
  *       Date & Time:    2008-Jan-27 15:49:51 (2008-Jan-27 23:49:51 GMT)
  *------------------------------------------------------------------------

  *       Design: dtmf_chip
  *
  *       Liberty Libraries used:
  *               /sev//DTMF_CHIP/TIMING/max/scmetropmk_cmos10lp_rvt_ss_1p08v_125c.lib
  *               /sev/DTMF_CHIP/TIMING/max/scmetropmk_cmos10lp_rvt_ss_0p8v_1p08v_125c.lib
  *               /sev/DTMF_CHIP/TIMING/max/scmetropmk_cmos10lp_rvt_ss_0p8v_125c.lib
  *               /sev/DTMF_CHIP/TIMING/max/scmetropmk_cmos10lp_lvt_ss_1p08v_125c.lib
  *               /sevDTMF_CHIP/TIMING/max/scmetropmk_cmos10lp_lvt_ss_0p8v_1p08v_125c.lib
  *               /sev/DTMF_CHIP/TIMING/max/scmetropmk_cmos10lp_lvt_ss_0p8v_125c.lib
  *               /sev/DTMF_CHIP/TIMING/max/scmetropmk_cmos10lp_hvt_ss_1p08v_125c.lib
  *               /sev/DTMF_CHIP/TIMING/max/scmetropmk_cmos10lp_hvt_ss_0p8v_1p08v_125c.lib
  *               /sev/DTMF_CHIP/TIMING/max/scmetropmk_cmos10lp_hvt_ss_0p8v_125c.lib
  *    /sev/DTMF_CHIP/TIMING/max/scmetro_cmos10lp_rvt_ss_1p08v_125c.lib
  *    /sev/DTMF_CHIP/TIMING/max/scmetro_cmos10lp_rvt_ss_0p8v_125c.lib
  *    /sev/DTMF_CHIP/TIMING/max/scmetro_cmos10lp_lvt_ss_1p08v_125c.lib
  *    /sev/DTMF_CHIP/TIMING/max/scmetro_cmos10lp_lvt_ss_0p8v_125c.lib
  *    /sev/DTMF_CHIP/TIMING/max/scmetro_cmos10lp_hvt_ss_1p08v_125c.lib
  *     /sev/DTMF_CHIP/TIMING/max/scmetro_cmos10lp_hvt_ss_0p8v_125c.lib
  *     /sev/TIMING/max/rom_512x16A_ss_1v08_125c_syn.lib
  ```

```
*     /sev/DTMF_CHIP/TIMING/max/pllclk_slow.lib
*               /sev/DTMF_CHIP/TIMING/max/iometroil_cmos10lp_hvt_ss_1p08v_2p3v_2p5v_125c.lib
*               /sev/DTMF_CHIP/TIMING/max/iometroil_cmos10lp_hvt_ss_0p8v_2p3v_2p5v_125c.lib
*               /sev/DTMF_CHIP/TIMING/min/scmetropmk_cmos10lp_rvt_ff_1p32v_m40c.lib
*               /sev/DTMF_CHIP/TIMING/min/scmetropmk_cmos10lp_rvt_ff_1p1v_1p32v_m40c.lib
*               /sev/DTMF_CHIP/TIMING/min/scmetropmk_cmos10lp_lvt_ff_1p32v_m40c.lib
*               /sev/DTMF_CHIP/TIMING/min/scmetropmk_cmos10lp_lvt_ff_1p1v_1p32v_m40c.lib
*               /sev/DTMF_CHIP/TIMING/min/scmetropmk_cmos10lp_hvt_ff_1p32v_m40c.lib
*               /sev/DTMF_CHIP/TIMING/min/scmetropmk_cmos10lp_hvt_ff_1p1v_1p32v_m40c.lib
*

/sev/DTMF_CHIP/TIMING/min/scmetro_cmos10lp_rvt_ff_1p32v_m40c.lib
*
/sev/DTMF_CHIP/TIMING/min/scmetro_cmos10lp_lvt_ff_1p32v_m40c.lib
*
/sev/DTMF_CHIP/TIMING/min/scmetro_cmos10lp_hvt_ff_1p32v_m40c.lib
*
/sev/DTMF_CHIP/TIMING/min/rom_512x16A_ff_1v32_m40c_syn.lib
*
/sev/DTMF_CHIP/TIMING/min/pllclk_fast.lib
*               /sev/DTMF_CHIP/TIMING/min/iometroil_cmos10lp_hvt_ff_1p32v_2p7v_2p5v_m40c.lib
*               /sev/DTMF_CHIP/TIMING/min/iometroil_cmos10lp_hvt_ff_1p1v_2p7v_2p5v_m40c.lib
*
*       Power Domain used:
*               Rail:     VDD_AO        Voltage:       1.32
*               Rail: VDD_TDSPCore      Voltage:       1.32
*               Rail: VDD_TDSPCore_R    Voltage:       1.32
*
*       Parasitic Files used:
* /sev/DTMF_CHIP/SPEF/dtmf_chip.decoup.spef
*
*       DEF Files used:
* /sev/DTMF_CHIP/DEF/dtmf_chip.def
*
*       Power Units = 1mW
*       Time Units = 1e-09 secs
*       report_power
--------------------------------------------------------------------------------
  Total Power
--------------------------------------------------------------------------------
Total Internal Power:      0.01943(62.6774%)
Total Switching Power:     0.01057(34.0968%)
Total Leakage Power:       0.001(3.2258%)
Total Power:               0.031
--------------------------------------------------------------------------------
Group      Internal      Switching     Leakage       Total  Percentage
Power      Power         Power         Power  (%)
--------------------------------------------------------------------------------
Sequential    0.0006738   0.0002215   4.045e-05   0.0009358     3.019
Macro         0.0004916   1.515e-05   0.0002609   0.0007677     2.477
IO            0.01743     0.009085    0.0003172   0.02683      86.56
Combinational 0.0008324   0.001247    0.0003815   0.002461      7.941
--------------------------------------------------------------------------------
Total 0.01943     0.01057       0.001       0.031         100
```

```
--------------------------------------------------------------------------------
Rail                  Voltage   Internal   Switching   Leakage Total  Percentage                        Power     Power      Power
       Power  (%)
--------------------------------------------------------------------------------
VDD_AO                 1.32  0.0008238   0.0003404   0.0002694    0.001434       4.625
VDD_TDSPCore           1.32   0.001126    0.001022   0.0003931    0.002541       8.197
VDD_TDSPCore_R         1.32  5.833e-06   3.253e-07   8.371e-07   6.996e-06     0.02257


Clock                    Internal   Switching     Leakage       Total  Percentage
                             Power      Power         Power         Power  (%)
--------------------------------------------------------------------------------

m_digit_clk                 0           0           0           0           0
m_ram_clk                   0           0           0           0           0
m_dsram_clk                 0           0           0           0           0
m_spi_clk           1.018e-06    1.09e-06   2.572e-08   2.134e-06    0.006884
m_rcc_clk           1.282e-05   1.513e-05   2.672e-07   2.821e-05     0.09103
m_clk               5.039e-05   7.278e-05   7.658e-07   0.0001239      0.3998
refclk              1.11e-05   3.017e-05   2.488e-07   4.152e-05       0.134
--------------------------------------------------------------------------------
Total               7.533e-05   0.0001192   1.307e-06   0.0001958      0.6317
--------------------------------------------------------------------------------

*       Power Distribution Summary:
*  Highest Average Power:     IOPADS_INST/Prefclkip (PBCSCT2B):     0.003727
*  Highest Leakage Power: DTMF_INST/ROM_512x16_0_INST (rom_512x16A): 8.899e-05
*               Total Cap:      2.46221e-10 F
*               Total instances in design:  8525
*               Total instances in design with no power:    24
*                 Total instances in design with no activty:    24
*               Total Fillers and Decap:     8
--------------------------------------------------------------------------------
```

- Next command creates a leakage report (see example report below):

```
report_power -leakage
*--------------------------------------------------------------------------------
*    Tempus Timing Signoff Solution 08.10-b050_1 (64bit) 08/21/2008 19:23 (Linux 2.6)
*
*
*       Date & Time:    2008-Aug-24 21:34:17 (2008-Aug-25 04:34:17 GMT)
*
*--------------------------------------------------------------------------------
*
*       Design: dma_mac
*
*       Liberty Libraries used:
*               DATA/timing_libs//tcbn65lp_LVL_c060217wc0d90d9.lib
*               DATA/timing_libs//tcbn65lpcgwc.lib
*               DATA/timing_libs//tcbn65lplvtwc.lib
*               DATA/timing_libs//tcbn65lpwc.lib
*               DATA/timing_libs//tsdn65lpa1024x32m8f_100a_wc.lib
*               DATA/timing_libs//tsdn65lpa128x16m8f_100a_wc.lib
*               DATA/timing_libs//tcbn65lpcgwc0d72.lib
*               DATA/timing_libs//tcbn65lplvtwc0d72.lib
```

```
*               DATA/timing_libs//tcbn65lpwc0d72.lib
*               DATA/timing_libs//tsdn65lpa1024x32m8f_0p84v_wc.lib
*               DATA/timing_libs//tsdn65lpa128x16m8f_0p84v_wc.lib
*
*       Power Domain used:
*               Rail:      VDDlu        Voltage:       0.84
*               Rail:       VDDm        Voltage:       1.08
*               Rail:        VDD        Voltage:       1.08
*
*       Power Units = 1mW
*
*       Time Units = 1e-09 secs
*
*       report_power -out_file Leakage.rep -leakage
*
--------------------------------------------------------------------------------
Total Power
--------------------------------------------------------------------------------
Total Leakage Power:       0.1879
--------------------------------------------------------------------------------
Group                           Leakage       Percentage
                                Power         (%)
--------------------------------------------------------------------------------
Sequential                        0.0127       6.756
Macro                               0.1556        82.8
IO                                     0           0
Combinational                     0.01962       10.44
--------------------------------------------------------------------------------
Total                               0.1879         100
--------------------------------------------------------------------------------
Rail                  Voltage   Leakage       Percentage
                                Power         (%)
--------------------------------------------------------------------------------
VDDau                       0          0           0
VDDlu                    0.84    0.06477       34.47
VDDm                     1.08   0.003587       1.909
VDD                      1.08     0.1195       63.62

Clock                           Leakage       Percentage
                                Power         (%)
--------------------------------------------------------------------------------
phy_rxclk_2                            0           0
phy_txclk_2                    0.0004786      0.2547
phy_rxclk_1                            0           0
phy_txclk_1                     3.39e-05     0.01804
clk                             0.001723      0.9171
--------------------------------------------------------------------------------
Total                           0.002236        1.19
--------------------------------------------------------------------------------


--------------------------------------------------------------------------------

*       Power Distribution Summary:
```

```
*               Highest Average Power: ethernet_mac_2/tx_fifo_module_from_dma/fifo1/dual_port_
   ram_4_tx_fifo/ram2P1024x32/ram (TSDN65LPA1024X32M8F):      0.02424
*               Highest Leakage Power: ethernet_mac_2/tx_fifo_module_from_dma/fifo1/dual_port_
   ram_4_tx_fifo/ram2P1024x32/ram (TSDN65LPA1024X32M8F):      0.02424
*               Total Cap:      1.01625e-10 F
*               Total instances in design: 29235
*               Total instances in design with no power:  3596
*          Total instances in design with no activty:   115
*               Total Fillers and Decap:     0
--------------------------------------------------------------------------------

Total leakage power = 0.187903 mW
  Cell usage statistics:
  Library tsdn65lpa128x16m8f_1v08, 1 cells ( 0.003900%) , 0.01016 mW (    5.407063%  )
  Library tsdn65lpa1024x32m8f_1v08, 6 cells ( 0.023402%), 0.145427 mW (    77.394568%  )
  Library tcbn65lp_LVL_c060217wc0d90d9 , 120 cells ( 0.468037%), 0.000175605 mW    ( 0.093455%  )

  Library unknown , 3881 cells ( 15.137096%) , 0.00630072 mW ( 3.353178%  )
  Library tcbn65lpcgwc , 118 cells ( 0.460236%) , 0.000625092 mW ( 0.332667%  )
  Library tcbn65lplvtwc , 3762 cells ( 14.672959%) , 0.0110529 mW ( 5.882233%  )
  Library tcbn65lpwc , 17751 cells ( 69.234370%) , 0.0141619 mW ( 7.536835%  )
```

Related Information

- power Category Attributes
- "Static Power, IRdrop and EM Analysis" in the *Voltus Stylus Common UI User Guide*
- "Dynamic Power and IRDrop Analysis" in the *Voltus Stylus Common UI User Guide*

report_unannotated_nets

```
report_unannotated_nets
[-help]
[-file filename]
-type annotationType
```

Shows the percentage of nets annotated by a specific type and also prints the names of all nets that are not annotated by that type. Annotation information is stored on a per-file basis for activity, VCD, TCF, and user commands to provide detailed annotation reports when requested.

`report_unannotated_nets` can be invoked multiple times with different report types (`-type`). You must specify this command before `report_power`.

Parameters

`-file filename`	Specifies the text file name that includes the annotation report information. If no file is specified, the report will be written to the default output file `annotation_type.log`, where *type* is taken from the `-type` argument specified. *Data_type*: string, optional
`-help`	Outputs the command usage and a brief description about the command parameters.
`-type annotationType`	The type can be any of the following: • `all` Shows annotation reports for all types. • `activity` Shows annotation data for all types of activity specifications including TCF, VCD, SDC, TWF, and user commands that set activity. • `cap` Shows annotations from the SPEF files. • `fsdb` Shows annotations from the FSDB files. • `saif` Shows annotations from the SAIF files. • `tcf` Shows annotations from the TCF files. • `timing` Shows annotations from the TWF files. This type is applicable only to the dynamic power analysis flow. • `vcd` Shows annotation from the VCD files. *Data_type*: string, optional

Example

- The following command dumps the nets in the netlist that are missing in the vcd file:
  ```
  report_unannotated_nets -type vcd -file vcd.log
  ```
  The net names are written in the file `vcd`.log.
- The following command dumps the nets in the netlist which do not have an activity annotated from any source (VCD/TCF/SDC/TWF/user defined) and writes the net names in the file `unannotatednets.log`:

```
report_unannotated_nets -type all -file unannotatednets.log
```

Related Information

- read_activity_file
- "Dynamic Power and IRDrop Analysis" in the *Voltus Stylus Common UI User Guide*

reset_power_activity

`reset_power_activity`

Resets all activity in the database. You can specify this command to reset all activity and power information in a given session.

> You can reset activity from an activity file using `read_activity_file -reset`, reset activity for nets, pins, and ports using `set_switching_activity -reset`, and reset the default switching activities using `set_default_switching_activity -reset`.

Parameters

None

Related Information

- `read_activity_file`
- `set_default_switching_activity`
- `set_switching_activity`
- "Dynamic Power and IRDrop Analysis" in the *Voltus Stylus Common UI User Guide*

set_default_switching_activity

```
set_default_switching_activity
[-black_box_density value]
[-black_box_duty value]
[-duty value]
[-global_activity factor]
[-hierarchy hierarchy_name]
[-input_activity factor]
[-period value]
[-sequential_activity factor]
[-reset]
[-reset_type { global_activity | seq_activity ... }]
[-clock_gates_enable {activity_factor}]
[-integrated_clock_gate_ratio num]
[-combinational_clock_gate_ratio num]
[-clock_gates_output_ratio num]
[-clip_activity_to_domain_freq {true | false}]
[-clock_gates_output {activity_factor}]
[-name activity_name]
[-macro_activity factor]
```

Specifies the switching activity for all primary inputs, nets, and other devices in the design whose activity has not been previously defined through user attributes, the toggle count format (TCF) file, the value change dump (VCD) file, or the tracing of the clock network.

You can use this command to specify user-defined activities at the design level.

Parameters

`-black_box_density value`	
	Specifies to set the default transition density on all the output pins of all blackboxes in the design. The `-black_box_density` and `-black_box_duty` parameters must be used together. If one or both of these parameters are not specified, then the average transition density from all the input pins of the blackboxes.
`-black_box_duty value`	
	Specifies to set the default duty on all the output pins of all blackboxes in the design. The `-black_box_density` and `-black_box_duty` parameters must be used together. If one or both of these parameters are not specified, then the average transition density from all the input pins of the blackboxes.
`-clip_activity_to_domain_freq {true | false}`	
	Specifies to perform domain-based activity clipping. This parameter clips data paths to 1x frequency whereas the clock networks are still at 2x f allows you to have more control to determine the rate of clipping with respect to frequency. *Data_type*: string, optional
`-clock_gates_enable {activity_factor}`	
	Specifies the activity at the enable pin(s) of the clock gating cells. *Data_type*: string, optional
`-clock_gates_output {activity_factor}`	

	Specifies the activity at the output pin(s) of the clock gating cells. This works for both ICGs as well as combinational clock gating cells. **Example**: `set_default_switching_activity -global_activity 0.02 -clock_gates_output 2.0` (clock is transparent) transition_density of clock gate output is calculated by: `(1/clock_domain_frequency) * clockgateOutputsActivity = 1/(0.431*e-9) * activity_factor` In this case, transition_density = `clock_domain_frequency * clockgateOutputsActivity` = 1e9/0.431 * 2 = 4,640,371,200 *Data_type*: string, optional
`-clock_gates_output_ratio` *num*	

	Specifies to control transition density at the output pin(s) of the integrated and combinational clock gating cells. This parameter is equivalent to `-integrated_clock_gate_ratio` and `-combinational_clock_gate_ratio` together with the same value.
	For example, the `set_default_switching_activity -clock_gates_output_ratio 0.6` is equivalent to `set_default_switching_activity -inte 0.6 -combinational_clock_gate_ratio 0.6`.
	Note: This parameter has a cascading effect when there are multiple levels of ICG cells on the clock path. For example, when `-clock_gates_ou` the first level ICG cell will have 0.5 ratio of the root clock's toggle density, the second level ICG cell will have 0.5*0.5=`0.25` ratio of the root clock third level ICG cell will have 0.5*0.5*0.5=`0.125` ratio of the root clock's toggle density, and so on.
	Note: For dynamic power analysis, fastest clock on the path is used for calculating output toggle density. However, for static power analysis, w toggle densities is used.
	Example for static power analysis:
	`set_default_switching_activity -global_activity 0.02 -clock_gates_output_ratio 1.0`
	Transition density for either a combinational or an integrated clock gating cell with 'N' inputs is calculated by:
	`output_pin_transition_density = (summation) weighted_input_transition_density = clock_gates_output_ratio * [(duty_input_pin1 (duty_input_pin2 * Td_input_pin2) + ... + (duty_input_pinN * Td_input_pinN)]`
	Given a 2:1 clock Mux and assuming duty of 0.5 for all input pins:
	Td is the toggle density for each input pin of the instance `(Td_pinSel , Td_pinI0 , Td_pinI1).`
	`output_pin_transition_density = 1.0 * [0.5 * Td_pinSel + 0.5 * Td_pinI0 + 0.5 * Td_pinI1]`
	Data_type: string, optional

`-combinational_clock_gate_ratio` *num*	
	Specifies to set the propagation ratio for combinational clock gate cell outputs. When specified, this parameter sets the output activity of any ins combinational clock gating cell to *num* times the fastest transition density arriving on the instance's clock inputs. This parameter cannot be spec parameter. **Notes:** • This parameter does not apply to combinational clockgates with a constant data input, that is, a select pin tied low. • For dynamic power analysis, fastest clock on the path is used for calculating output toggle density. However, for static power analysis, weig toggle densities is used. For a detailed static power analysis example, refer to '`-clock_gates_output_ratio`', limiting calculations only to co
`-duty` *value*	Specifies the duty cycle, which is the probability that the signal is a logical 1. **Note:** You must specify the `-period` parameter when assigning a duty. *Default*: `0.5` (equivalent to a 50 percent duty cycle) *Data_type*: string, optional
`-global_activity` *factor*	
	Specifies the average number of times that all unset nodes switch in a clock cycle. If not specified the fastest domain frequency will be used un domains. **Note**: Due to activity precedence, the `-global_activity` and `-sequential_activity` parameters of the `set_default_switching_activity` comn exclusive. If both the parameters are specified, `-global_activity` takes precedence. *Data_type*: string, optional
`-hierarchy` *hierarchy_name*	
	A hierarchical name that the global activity is being assigned to. If not provided, the whole design is assumed. *Data_type*: string, optional

`-integrated_clock_gate_ratio` *num*	
	Specifies to set the propagation ratio for integrated clock gate (ICG) cell outputs. When specified, this parameter sets the output pin of all identif times the fastest transition density arriving on clock inputs of the instance. *Data_type*: string, optional This parameter cannot be specified with the `-hierarchy` parameter.
`-input_activity` *factor*	
	Specifies the average number of times that a primary input switches in a clock cycle. It can be any positive number. *Default*: `0.2` (Switches once every five clock cycles.) *Data_type*: string, optional
`-macro_activity` *factor*	
	Specifies the activity factor for macros in a design. The allowed values for this parameter are between `0` to `1`. If you set the macro activity to `.30`, 30% of macros in the design would be selected for switching in a given cycle during dynamic simulation. Wh specified for the static power analysis, all output pins of the macros are assigned the specified activity. *Default*: `0.2` *Data_type*: string, optional
`-name` *activity_name*	

	Allows you to set the activity level for the specified jitter activity name. You can define a low/high activity level for jitter analysis.
	Examples:
	`set_default_switching_activity -name low -clock_gates_output 0 -global_activity 0.01`
	`set_default_switching_activity -name high -clock_gates_output 2 -global_activity 0.2`
	Data_type: string, optional

`-period` *value*	Specifies the default operating period (1/frequency). It is the period that the activity is referenced to. If not specified, the software uses the *domi* The specified period will only be applied to nets that do not have reference clock (ref_clock) associated with the net. If no clock is specified, the software generates an error and prompts you to enter one. Units in seconds (`s`), milliseconds (`ms`), microseconds (`us`) picoseconds (`ps`). **Note:** If you enter the period as a negative number, the software reverts back to using the default frequency. *Default*: The time unit is defined in the first `.lib` file read during design import. *Data_type*: string, optional
`-sequential_activity` *factor*	
	Specifies the activity of outputs of the sequential logic. This value will be used at the output ports of flops for propagating activity. If the sequential activity is not specified, propagation will be done for the full design starting from the input ports. *Data_type*: string, optional
`-reset`	Resets all specified options back to default values. *Data_type*: bool, optional
`-reset_type { global_activity | seq_activity ... }`	
	Specifies to selectively reset the global specifications. **Note:** You must use the `-reset_type` parameter in conjunction with the `-reset` parameter. *Data_type*: string, optional

Activity Precedence

See "Activity Precedence" in `read_activity_file`.

Examples

- The following command specifies that the primary inputs in the design will switch once every ten clock cycles, with a clock period of 7520 ns and a duty cycle of 50 percent:
  ```
  set_default_switching_activity -input_activity 0.10 -period 7520ns -duty .5
  ```
- The following command specifies that all unswitched nodes in the design will switch once every ten clock cycles, with a clock period of 7520 ns and a duty cycle of 50 percent:
  ```
  set_default_switching_activity -global_activity 0.10 -period 7520ns -duty .5
  ```
- The next example defines the switching activity, sequential activity, and the period:
  ```
  set_default_switching_activity -input_activity 0.2 -period 10 \
      -sequential_activity 0.1
  ```
- The next example sets the default global activity to 0.5 for the hierarchy dma_dut:
  ```
  set_default_switching_activity -global_activity 0.5 -hierarchy dma_dut
  ```
- The following command resets the sequential and global activity specification:
  ```
  set_default_switching_activity -reset -reset_type
  {sequential_activity,global_activity}
  ```

Related Information

- read_activity_file
- set_switching_activity
- "Dynamic Power and IRDrop Analysis" in the *Voltus Stylus Common UI User Guide*

set_dynamic_power_simulation

```
set_dynamic_power_simulation
[-period value]
[-resolution value]
[-reset]
[-activity_pattern {time1 activity_name1 time2 activity_name2 .... }]
```

Specifies the period and resolution for a dynamic power simulation. This command is used to specify the simulation period in the dynamic vectorbased/vectorless and power-up flows.

Parameters

-activity_pattern {*time1 activity_name1 time2 activity_name2 .... }*	
	Divides the total simulation time based on the windows given in the `activity_pattern` parameter. For a particular window, it uses the defined activity specified with the `set_default_switching_activity` command. When the user-defined low cycle and high cycle time is specified for the input/sequential activity, all the clocks will be in the low or high cycle until completely past the cycle transition time. Following is the use model of this parameter: • `set_default_switching_activity` **`-name low -input_activity 0.05 -sequential_activity 0.1`** `-clock_gates_output 0.2` • `set_default_switching_activity` **`-name high -input_activity 0.1 -sequential_activity 0.2`** `-clock_gates_output 2.0` • `set_dynamic_power_simulation -period 15ns -resolution 50ps` **`-activity_pattern {0 high 5ns low 10ns high}`** Here, high activity is used for 0ns to 5ns, low activity is used for 5ns to 10ns, and high activity is used for 10ns to 15ns. *Data_type*: string, optional
`-period` *value*	Specifies the simulation period that dynamic power analysis will use. The supported time units are `s`, `ms`, `us`, `ns`, and `ps`. If no explicit time unit is specified, the default `ns` (nanosecond) will be applied. *Data_type*: string, optional
`-reset`	Resets all specified options back to default values. *Data_type*: bool, optional
`-resolution` *value*	Specifies the transient time step size or resolution. The supported time units are `s`, `ms`, `us`, `ns`, and `ps`. If no explicit time unit is specified, the default `ps` (picosecond) will be applied. *Data_type*: string, optional

Examples

- The following command sets up the period and resolution for a dynamic simulation:
  ```
  set_dynamic_power_simulation -period 5ns -resolution 100ps
  ```

Related Information

- `power Category Attributes`
- `set_default_switching_activity`
- "Dynamic Power and IRDrop Analysis" in the *Voltus Stylus Common UI User Guide*

set_instance_temperature_file

```
set_instance_temperature_file
file
[-reset]
```

Specifies the instance temperature file to support thermal-aware leakage. You can use this command to do thermal-aware power calculation. Based on the temperature specified in the file, the software will calculate temperature and leakage power.

In the presence of a trilib set, the internal and leakage power will be interpolated as the per the given temperature specification.

Parameters

file	Specifies the name of the instance temperature file. *Data_type*: string, optional
-reset	Resets the temperature file. *Data_type*: bool, optional

Related Information

- report_power
- power Category Attributes
- "Static Power, IRdrop and EM Analysis" in the *Voltus Stylus Common UI User Guide*
- "Dynamic Power and IRDrop Analysis" in the *Voltus Stylus Common UI User Guide*

set_power

```
set_power
[value]
[-cap value]
[-fanout_limit value]
[-reset]
[-leakage value]
[-pg_pin {pg_pin}]
[-pg_net railname]
[-pwl_waveform]
[-sticky]
[-dynamic_async_mode { mode_name }]
[-dynamic_switch_event { switching_events }]
[-dynamic_switch_pattern pattern]
[-scale_factor value}
[-include_ddv_modes {pattern_list}]
[-internal]
[-switching value]
[-clocks {all|clock_name}]
[-exclude_clocks]
[-exclude_ddv_modes {pattern_list}]
[-base_cells cell_name value]
[-insts inst_list value]
[-pin pin_name pin_power]
[-force]
[-custom_macro_pwl_waveform filename]
[-repeat]
[-ascii_power_file filename]
[-trigger_time_adjustment time]
[-no_propagation]
[-static_mode { custom_label_name | mode_name }]
[-slew value]
[-signal_net netname]
[-default_dynamic_pgv_mode { mode_name }]
```

Specifies the amount of power for all or part of the design. This command is used to define user-specified power on cells and instances. You must specify this command before `report_power`.

The `set_power` command can accept a cell/instance list using the `get_cells`/`get_lib_cells` commands. You can use the `get*` commands to create a collection of cells/instances whose name matches the supplied pattern list. Therefore, providing you with the flexibility in providing lists through use of regular filter patterns. This feature is supported in both static and dynamic analysis engines.

Parameters

`-ascii_power_file` *filename*	
	Specifies the name of an instance list file. This file contains the list of instances in the following format: *instance_name power_value <pin_name>* Here, • *instance_name* is the instance name to which power is to be distributed. • *power_value* is the power value for the instance in watts. • *pin_name* is an optional argument to specify the pin name. When specified, allows you to assign power to the specific pin • The three fields should be in one line, and each instance should be in a new line. *Data_type*: string, optional
`-base_cells` *cell_name value*	

	Specifies the cell name. If the cell or instance name is not specified, the power will be distributed to all rails. If you want to specify power for a group of cells/instances, use the `get_cells`/`get_lib_cells` commands. The `-base_cells` p commands to create a collection of cells/instances whose name matches the supplied pattern list. For example, `-base_cell 2mw` *Data_type*: string, optional
`-cap value`	
	Specifies the capacitance value to all instances with nets more than or equal to the fanout limit (`-fanout_limit`). *Data_type*: string, optional
`-clocks {all|clock_name}`	
	Specifies the power value for clock networks. You can specify the power value for a specific clock network or the clock netw whole design. **Note:** This parameter cannot be used with the `-base_cells`, `-insts` and `-pg_net` parameters. *Data_type*: string, optional
`-custom_macro_pwl_waveform filename`	
	Specifies to use a custom trigger file to apply instance or cell PWL waveforms. Given one master cell PWL waveform, the so generate scaled PWL waveforms of the other companion cells defined in the trigger file proportional to the cell internal powe **Note**: This parameter must be used with `set_power -dynamic_switch_pattern` in the vectorless flow. For more information on the trigger file format, see Trigger File Format. *Data_type*: string, optional
`-default_dynamic_pgv_mode { mode_name }`	
	Specifies the current characterization mode to be used during dynamic power and jitter analyses. This parameter can be sp with multiple functional modes. When this parameter is specified, the Power Calculation engine uses this preferred mode in scheduling the switching event saves these events in the trigger file. The use model of this parameter is: `set_power -base_cells {} | -insts {} -default_dynamic_pgv_mode { mode_name }` This parameter is mutually exclusive to the `-dynamic_switch_pattern` parameter. If a cell or instance has both the `-dynamic_ preferred_ddv_mode` parameters defined in the same Tcl script, `-dynamic_switch_pattern` takes higher priority. *Data_type*: string, optional
`-dynamic_switch_event { switching_events }`	
	Specifies the switching events for cells, instances, and pins along with their rise and fall time. `switching_events` is a string "r" or "f", where "r" is for rise toggle and "f" is for fall toggle. The `-dynamic_switch_event` parameter should not be specified w
`-dynamic_async_mode { mode_name }`	

	Specifies to trigger asynchronous macros based on the length of the PWL waveform to create a single continuous waveform you to specify the asynchronous mode for an instance or a cell. The start and end of the PWL saved in the PGV must have s order to avoid sudden jump in the current.
	Use model:
	`set_power -dynamic_async_mode {mode} [-insts {name} | -base_cells {name}]`
	Example:
	`set_power -base_cells pll -dynamic_asynchronous_mode {READ_C}`
	Here, power analysis will trigger the `READ_C` mode and automatically splice the waveform based on its length.

`-dynamic_switch_pattern pattern`	

	Specifies stimulus pattern on the pins of an instance and cell. Specify this parameter only in the dynamic mode for the type : The *pattern* is applied to all pins (input and output) on the instance/cell. It is a string. *pattern* specifies the switching pattern. If there is more than one clock domain with timing windows on a pin, only the windo be used, and the pattern will be applied to that clock period. It is a string of 0, 1, r, and f values. 0 for cycles not to switch and for rise only and f for fall only. The pattern string should include enough digits to cover all of the clock cycles in the simulation period. If there are not enou padded with 0 values at the end. If there are too many, the pattern will be truncated to the number of applicable cycles. Tran the center of each applicable timing window. **Notes** • 0 and 1 is supported only in the probability-based vectorless flow. It is not supported in the state-propagation-based vecto based flow. • In case of macros, you can specify the mode names, that is, which mode switches in which clock cycle -dynamic_switch_ - }. You can specify "-" to indicate that instances will not switch in a particular clock cycle. If "-" is the first value in the string of v you must insert a space before "-", as shown below: • no switching: -dynamic_switch_pattern { - - } • no switching in the 1st clock period: -dynamic_switch_pattern { - r } Examples: In the following example, instance will not switch in the 4th clock cycle: set_power -dynamic_switch_pattern { f r f - r } In the following example, the mode write will switch in the 1st clock cycle, the mode read will switch in the 2nd and 4th cloc modes will switch in the 3rd clock cycle: set_power -dynamic_switch_pattern { write read - read } **Note**: Wildcards are supported only for -dynamic_switch_pattern with -pin. When -dynamic_switch_pattern is applied for without the -pin parameter, wildcards are not supported. *Data_type*: string, optional
-exclude_clocks	Excludes clock networks when specifying a scale factor or power specification for the whole design. This parameter cannot be used with the -base_cells, -insts and -pg_net parameters. *Data_type*: string, optional
-exclude_ddv_modes {*pattern_list*}	
	Excludes the current characterization (or Dynamic Detailed View) modes that match with the specified patterns. This param the modes to be switched, instead of switching all the current characterization modes defined in the power-grid view. **Use Model** set_power -insts|-base_cells *name* -include_ddv_modes {*pattern_list*} Here: • *pattern_list* is a list of mode names, separated by space or comma • the instance, cell, and pattern names can contain wildcard characters *Data_type*: string, optional
-fanout_limit *value*	
	Specifies the fanout limit for nets. If an instance has nets more than or equal to the fanout limit, its switching/internal/leakage When this parameter is specified, the user-defined power value is applied to all the driver instances that have nets more tha limit. *Data_type*: string, optional
-force	

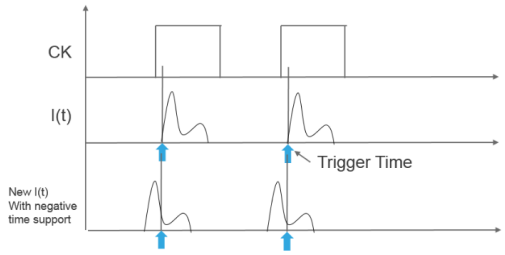
	Specifies to apply the `set_power` command even if there are Verilog mismatches. For dynamic or DEF-based static analysis to apply `set_power` to the DEF instances that do not exist in Verilog. *Data_type*: bool, optional
`-help`	Outputs the command usage.
`-include_ddv_modes {`*`pattern_list`*`}`	
	Includes only the current characterization (or DDV) modes that match with the specified patterns. This parameter allows you switched, instead of switching all the current characterization modes defined in the power-grid view. **Use Model** `set_power -insts|-base_cells` *`name`* `-include_ddv_modes {`*`pattern_list`*`}` Here: • *`pattern_list`* is a list of mode names, separated by space or comma • the instance, cell, and pattern names can contain wildcard characters *Data_type*: string, optional
`-internal`	Specifies the target internal power of the design, cell, or instance. To specify internal power for cells and instances, you can parameter in conjunction with the `-base_cells` or `-insts` parameter. *Data_type*: bool, optional
`-insts` *`inst_listvalue`*	
	Specifies the instance name. If the cell or instance name is not specified, the power will be distributed to all rails. *Data_type*: string, optional
`-leakage`	Specifies the target leakage power of the design, cell, instance, or power net. To specify leakage power for cells and instances you can use `-leakage` option in conjunction with `-base_cells` or `-insts` op To specify leakage power of a certain power net in the design you can use `-pg_net` option along with `-leakage` option. *Data_type*: bool, optional
`-no_propagation`	
	Specifies that the user-defined switch pattern is applied only after state-propagation is complete. The use model of the comm `set_power -dynamic_switch_pattern {`*`pattern`*`} -no_propagation` The `-no_propagation` parameter must be specified with the `-dynamic_switch_pattern` parameter. When the `-no_propagati` the dynamic power engine performs state-propagation first, and then overrides the single instance switching event by the us pattern without any further propagation. *Data_type*: bool, optional
`-pg_net` *`railname`*	
	Optional. The specified power or pwl for the cell, instance or design is applied to the specified rail. When not specified, the p all the rails associated with the instance. This option is generally used for MSMV cells. When this option is not specified the behavior is different for pwl and power options. When not specified and power option is applied for the whole design, cell, or instance. When not specified and pwl option is used, the pwl is applied to every rail of *Data_type*: string, optional
`-pg_pin {`*`pg_pin`*`}`	
	Specifies the switch pattern for a specific PG pin of an instance. This parameter must be specified with the `-dynamic_switch` *Data_type*: string, optional
`-pin` *`pin_name pin_power`*	

	Specifies pin-wise power numbers for a specific instance. This parameter allows you to control the power assigned to powe instance. Following is the use model of this parameter:
	`set_power -insts <inst_name> -pin <VDDA> <pin_power>`
	`set_power -insts <inst_name> -pin <VDDB> <pin_power>`
	Data_type: string, optional

`-pwl_waveform`	Specifies that power argument (`power | pwl`) is a waveform rather than a power number. *Data_type*: bool, optional
`-repeat`	Repeats dynamic switch pattern of all the clock cycles in the simulation period. *Data_type*: bool, optional
`-reset`	This parameter causes all previous `set_power` commands to be ignored. Note: You can selectively reset the instance/cell specifications. In addition, you can also selectively reset the cell and instar of objects through the `get_cells` and `get_lib_cells` commands. *Data_type*: bool, optional
`-scale_factor value`	

	Scales the power value by the specified scale factor. This parameter allows you to set a user-defined scale factor in the trig dynamic current of the PGV.
	To modify the scale factor, use the `-reset` parameter with the `-scale_factor` parameter.
	Example:
	`set_power -leakage -reset  -scale_factor 5 -instance A`
	You can use the following power scaling commands in the power-driven optimization flow when running Voltus within the Ir
	• `set_power -scale_factor <value> -leakage` => scale the leakage power value by the specified scale factor
	• `set_power -scale_factor <value> -internal` => scale the internal power value by the specified scale factor
	• `set_power -scale_factor <value> -switching` => scale the switching power value by the specified scale factor
	• `set_power -scale_factor <value>` => scale the leakage, internal, and switching power values by the specified scale fac
	When specifying these commands in the optimization flow, you must not specify any other parameter of the `set_power` comn
	Optimization Flow Example:
	`set_db opt_power_effort high`
	`set_db opt_leakage_to_dynamic_ratio 0.5`
	`set_power -scale_factor 15 -internal`
	`set_power -scale_factor 20 -switching`
	`report_power`
	Data_type: string, optional

`-signal_net netname`	
	Specifies the signal net name for power assignment. When this parameter is specified, the driver instances connected to the assigned the user-defined power value. *Data_type*: string, optional
`-slew value`	
	Specifies the slew value to all instances with nets more than or equal to the fanout limit (`-fanout_limit`). *Data_type*: string, optional
`-static_mode { custom_label_name | mode_name }`	

	Specifies the mode name or the custom label name associated with a cell or instance for computing instance current from P voltage multi-frequency (MvMf) modes. This current is used to compute the instance's internal power, and its switching and components would be set to zero. The use model of this parameter is given below: `set_power -insts -static_mode READ` If you specify custom label of a static MvMf mode, currents would be taken from it directly. Alternatively, if only mode name is cell/instance, the software snaps to the nearest custom label defined within this mode, based on the instance's voltage and currents. *Data_type*: string, optional
`-sticky`	Applies the simulation based PWL waveform as is, to the specified cell/instance for dynamic power analysis. Use this option only if you have specified `-pwl_waveform`. *Data_type*: bool, optional
`-switching` *value*	
	Specifies the target switching power of the design, cell, or instance. To specify switching power for cells and instances, you parameter in conjunction with the `-base_cells` or `-insts` parameter. *Data_type*: bool, optional
`-trigger_time_adjustment` *time*	
	Specifies to adjust the trigger time of the current waveforms for the various functional modes (read/write/idle) of the custom r allows you to start the waveform before the trigger time (negative time support), as shown below:  The specified time will be applied to the computed trigger time for each mode in the trigger files generated by power analysi trigger time (from timing window/vector) is computed as 1ns, 2ns, and 3ns, and if you specify `set_power -trigger_time_adj` software will adjust the trigger time to .9ns, 1.9ns, and 2.9ns. *Data_type*: string, optional

value	Specifies power in milliwatts during static power calculation or piecewise linear (`-pwl`) current waveform, during dynamic pc instance. The command accepts a unit specification (e.g. 10w).
	The format of the PWL waveform is:
	`{time1 current1 time2 current2 ...time_n current_n}`
	Time is in ns and current in mA.
	Example:
	`set_power -pwl_waveform -pg_net {} -base_cells BUFX2MTL {0 0 10 0 10.1 5 10.5 0}`
	Here, at `0ns` its value is `0mA`, at `10ns` its value is `0mA`, at `10.1ns` its value is `5mA`, and so on.
	Data_type: string, optional

Examples

- The following command sets power for vdd rail to 10 milliwatts:
  ```
  set_power -pg_net vdd 10mw
  ```

- The following command sets a power value of .005 milliwatts on power net vdd for instance clkin_neg_L14_I1:
  ```
  set_power -pg_net vdd -insts Top/A/clkin_neg_L14_I1 .005
  ```
- The following command sets a power value of .005 milliwatts for any cell named INVX2 in the design:
  ```
  set_power -base_cells INVX2 .005mw
  ```
- The following command defines a piecewise linear waveform of {0 0 10 0 10.1 5 10.5 0} for the cell of type BUFX2MTL for all power rails.
  ```
  set_power -pwl_waveform -pg_net {} -base_cells BUFX2MTL {0 0 10 0 10.1 5 10.5 0}
  ```
- The following command specifies the leakage power for the cell DFFRX1.
  ```
  set_power -leakage -base_cells DFFRX1 10mw
  ```
- The following commands specify the target leakage power for power nets VDD1 and VDD2.
  ```
  set_power -leakage -pg_net VDD1 100mw
  set_power -leakage -pg_net VDD2 200mw
  ```
- The following command specifies the PWL description of current for the instance SEHD130_1024X32X1CM8. The -sticky option tells the software to use the specified PWL simulation based waveforms as is.
  ```
  set_power -pwl_waveform -insts SEHD130_1024X32X1CM8 -sticky {0 0 1 1 2 2 7 0}
  ```
- The following command scales the total power of the instance i_1 by a scale factor of 2:
  ```
  set_power -scale_factor 2 -insts i_1
  ```
- The following command sets the total power of the design to 20mw while keeping the clock network power the same:
  ```
  set_power 20mw -exclude_clocks {all}
  ```
- The following command assigns a switching power of 3mw to all instances of the cell AND1:
  ```
  set_power -base_cells AND1 -switching 3mw
  ```
- The following command assigns 2mw power to all cells that match the specified pattern:
  ```
  set_power -base_cells[get_lib_cells NAND*] 2mw
  ```
- The following command specifies that mode1 switches in the first cycle, mode2 switches in the third cycle and no switching in others:
  ```
  set_power -dynamic_switch_pattern { mode1 0 mode2 0 }
  ```
- The following command resets power of all cells that match the specified pattern:
  ```
  set_power -reset -base_cells[get_cells A/*]
  ```
- The following command does not schedule DDV mode `scan*`:
  ```
  set_power -base_cells { * } -exclude_ddv_modes { scan* }
  ```
- The following command does not schedule DDV mode `scan*` for instance "`A1`":
  ```
  set_power -insts { A1 } -exclude_ddv_modes { scan* }
  ```
- The following command only schedules DDV mode names with the `*read*` and `*write*` pattern:
  ```
  set_power -base_cells { * } -include_ddv_modes { *read*, *write* }
  ```
- The following commands specify the toggle time for the pin names Q7 and Q6:
  ```
  set_power -dynamic_switch_event { 1.1ns r 5.5ns f } -insts clk_b_i2 -pin Q7
  set_power -dynamic_switch_event { 2.2ns r 6.6ns f } -insts clk_b_i2 -pin Q6
  ```

Trigger File Format

For specifying a custom trigger file, you must use the `-custom_macro_pwl_waveform` parameter. The format of the trigger file is:

Trigger File Format	Description
#Cell names	
`MASTER CELL <cellname1>`	Name of the reference cell.
`COMPANION_CELLS <cellname1 cellname2 … >`	Name of the cells that will use scaled PWL from the master cell.
#Mode and conditional details for trigger	

MODE_NAME <*mode1*> CONDITIONAL_INPUT <conditional input statement for cellname1>	Mode name and conditional input statement for the cell. Conditional input of current for the given mode.
CONDITIONAL_PIN <pin_name> (rise|fall|both)	Conditional pin statement for the cell. Signal pin for which the tool should rise or fall.
CONDITIONAL_STIMULUS_FILE <usim1.txt>	Simulation vector detail in the given mode.
USER_PWL_FILE <pwl1.txt>	User-specified PWL file that contains information about the PWL waveforms (power/ground currents). For an example of a PWL file, see PWL Format Example.
END_MODE	End mode section
#End cell section	
END	End cell section

Example of a Custom Macro Power Trigger File

```
MASTER_CELL        SRAM_A1
COMPANION_CELLS    { SRAM_A2, SRAM_A3, SRAM_A4, SRAM_A5 }
      MODE_NAME WRITE
               CONDITIONAL_INPUT = ( !SD & !SLP & !CEB & !WEB )
               CONDITIONAL_PIN = CLK { RISE }
               USER_PWL_FILE write_rise.txt
       END_MODE
      MODE_NAME READ
               CONDITIONAL_INPUT = ( !SD & !SLP & !CEB & WEB )
               CONDITIONAL_PIN = CLK { RISE }
               USER_PWL_FILE read_rise.txt
       END_MODE
      ...
END
```

Example of "USER_PWL_FILE" (Multiple regions)

```
UNIT CURRENT mA
UNIT TIME ns
       NET VSS
               REGION { 0 0 0.1 -1 0.2 -2 0.3 -3 0.4 -10 0.5 -50 0.6 -40 0.7 -30 0.8 -20 1 0 2 0 3 0 3.9 0 4 -0.1 4.5 -0.2 5 -4 5.5 -3 6 -2
7 0 }
       NET VDD
               REGION { 0 0 0.1 1 0.2 2 0.3 3 0.4 10 0.5 50 0.6 40 0.7 30 0.8 20 1 0 2 0 3 0 3.9 0 4 0.1 4.5 0.2 5 4 5.5 3 6 2 7 0 }
       NET VDDM
               REGION { 0 0 0.1 1 0.2 2 0.3 3 0.4 10 0.5 50 0.6 40 0.7 30 0.8 20 1 0 2 0 3 0 3.9 0 4 0.1 4.5 0.2 5 4 5.5 3 6 2 7 0 }
```

Related Information

- power Category Attributes
- report_power
- "Static Power, IRdrop and EM Analysis" in the *Voltus Stylus Common UI User Guide*
- "Dynamic Power and IRDrop Analysis" in the *Voltus Stylus Common UI User Guide*

set_power_calculation_temperature

```
set_power_calculation_temperature
temperature
```

Sets the temperature for static power calculation.

In the presence of a trilib set, the internal and leakage power will be interpolated as the per the given temperature specification.

Parameters

temperature	Specifies the temperature for static power calculation. **Note:** The static power calculation software gets the temperature value of each instance from the following sources (in order of precedence): • Instance temperature file • `set_power_calculation_temperature` command • The current operating condition temperature

Related Information

- power Category Attributes
- "Static Power, IRdrop and EM Analysis" in the *Voltus Stylus Common UI User Guide*

set_power_include_file

```
set_power_include_file
file
[-reset]
```

Specifies a power analysis include file. You can use the `set_power_include_file` command to specify certain commands of the dynamic engine which do not have a Voltus equivalent. For more information on these commands, refer to the Power Analysis Options chapter.

The `set_power_include_file` command is required only if the `power_static_netlist` attribute is set to `def`.

Note: The `set_power_include_file` command is used to pass the legacy power analysis commands and `report_power` command options only in the dynamic mode (vectorbased and vectorless). If the `power_method` attribute is set to `static`, the `set_power_include_file` command is ignored.

Parameters

file	Name of the include file. This file will include all commands that can not be translated to Voltus commands. In addition, you can also include the `report_power` command options to generate different power reports. *Data_type*: string, optional
`-reset`	Resets all options back to default values. *Data_type*: bool, optional

Examples

- The following command specifies a power include file of `power.inc`:
  ```
  set_power_include_file power.inc
  ```
- The following example shows the contents of the `power.inc` file.
  # power.inc file must use the pre-defined design and power objects as "design" # and "ChipPwr" respectively. The power output object is not supported.
  ```
  design calCellsToIgnore BUFX1
  ChipPwr calRailVoltage VDD 1.2

  report_power -net -outfile Net.rep
  report_power -cell {INV*} -outfile Invpower.rep
  ```

Related Information

- power Category Attributes
- "Dynamic Power and IRDrop Analysis" in the *Voltus Stylus Common UI User Guide*

set_power_output_dir

```
set_power_output_dir dir
[-reset]
```

Specifies the name of the directory that power information will be output to.

Parameters

dir	Specifies the name of the directory in which power analysis output information is written. *Data_type*: string, optional
-reset	Resets all options back to default values. *Data_type*: bool, optional

Examples

- The following command specifies the output directory called `static_power_AvallOn` in which the power information is written.
  ```
  set_power_output_dir static_power_AVallOn
  ```

Related Information

- power Category Attributes
- "Static Power, IRdrop and EM Analysis" in the *Voltus Stylus Common UI User Guide*
- "Dynamic Power and IRDrop Analysis" in the *Voltus Stylus Common UI User Guide*

set_switching_activity

```
set_switching_activity
[-reset]
[-activity factor | -transition_density transition_density ]
[-clock clock_name]
[-duty value]
[-inst instance_name]
[-net net_name | -port port_name | -pin pin_name | -input_port port_name |-output_port port_name | -bidir_port port_name]
[-period value]
[-integrated_clock_gate_ratio num]
[-combinational_clock_gate_ratio num]
[-hierarchy hierarchy_name]
[-scale value]
[-cell cell_name]
[-force]
[-quiet]
[-unclocked]
```

Specifies activity for nets, pins, and ports. This command is used to specify user-defined activities at specific pins/nets/ports. You must specify this command before `report_power`.

The `set_switching_activity` command can accept a net/pin/port list using the `get_pins`/`get_ports`/`get_nets` commands. You can use the `get*` commands to create a collection of nets/pins/ports whose name matches the supplied pattern list. Therefore, providing you with the flexibility in providing lists through use of regular filter patterns. This feature is supported in both static and dynamic analysis engines.

Parameters

`-activity` *factor*	
	Specifies the activity for a net, pin, or top-level port. The *activity_factor* specifies the number of times the net, pin, or port switches in a clock cycle. Enter any positive number. If if the activity is set to 0.5, the activity happens once for every two clock cycles. If the activity is set to 2.0, the activity happens two times per clock cycle. There is no default value. *Data_type*: string, optional
`-bidir_port` *port_name*	
	Specifies the name of the bidirectional port. The port names are in the following format: bidirectional*-port*. This parameter supports wildcards (*). *Data_type*: string, optional
`-cell` *cell_name*	
	Specifies the activity on all output pins for all instances of the specified cell. *Data_type*: string, optional
`-clock` *clock_name*	
	Assigns global activity factor to all nets in domain of *clock_name* if and only if *clock_name* is the fastest clock in the domain set. To use this option, `-clock` and `-activity` must be used without any other options. *Data_type*: string, optional
`-combinational_clock_gate_ratio` *num*	
	Specifies to set the propagation ratio for combinational clock gate cell outputs. When specified, this parameter sets the output activity of any instance that is identified as a combinational clock gating cell to *num* times the fastest transition density arriving on the instance's clock inputs. You can specify this ratio for a net, pin, port, or clock. This parameter cannot be specified with the `-hierarchy` parameter. *Data_type*: string, optional
`-force`	

	Specifies to apply user-defined activities for specific nets, pins, and ports, without having to load a verilog netlist. When a verilog netlist is not specified, the `set_switching_activity` command does not support the `-cell` parameter, and the net/pin/port list using the `get_pins/get_ports/get_nets` commands. *Data_type*: bool, optional
`-transition_density` *transition_density*	
	Specifies the transition density for a specific net, pin, or top-level port. Transition density specifies the number of transitions per second. it can be any positive number. There is no default value. **Note:** If you specify the `-transition_density` parameter in conjunction with the `-period` parameter, the software issues a warning and the `-period` parameter is ignored. *Data_type*: string, optional
`-duty` *value*	
	Specifies the duty cycle, which is the probability that the signal is a logical 1. The `-period` parameter must be specified to add a duty. If not specified, the `set_switcing_activity` command picks this value from the external TWF or SDC through the Common Timing Engine (CTE). *Default*: Uses the duty specified by the `set_default_switching_activity` command if the `-duty` parameter is not specified. *Data_type*: string, optional
`-hierarchy` *hierarchy_name*	
	A hierarchical name that an activity is being assigned to. *Data_type*: string, optional
`-integrated_clock_gate_ratio` *num*	
	Specifies to set the propagation ratio for integrated clock gate (ICG) cell outputs. When specified, this parameter sets the output pin of all identified ICG instances to *num* times the fastest transition density arriving on clock inputs of the instance. You can specify this ratio for a net, pin, port, or clock. This parameter cannot be specified with the `-hierarchy` parameter. *Data_type*: string, optional
`-input_port` *port_name*	
	Specifies the name of the input port. The port names are in the following format: *input-port*. This parameter supports wildcards (*) *Data_type*: string, optional
`-inst` *instance_name*	
	Specifies the activity on a specific instance. *Data_type*: string, optional
`-net` *net_name*	
	Specifies the name of the net. This parameter supports wildcards (*). *Data_type*: string, optional
`-output_port` *port_name*	
	Specifies the name of the output port. The port names are in the following format: *output-port*. This parameter supports wildcards (*) *Data_type*: string, optional
`-period` *value*	

	Specifies the period that the activity is referenced to. If not specified, a default frequency is used. If a negative number is specified, the static power calculation engine reverts back to the default frequency. Units in seconds (`s`), milliseconds (`ms`), microseconds (`us`), nanoseconds (`ns`), or picoseconds (`ps`). **Note:** If you specify the `-period` parameter in conjunction with the `-transition_density` parameter, the software issues a warning and the `-period` parameter is ignored. *Default*: The time unit is defined in the first `.lib` file read during design import. If the time unit is not defined, the period specified in the `set_default_switching_activity` command is used. *Data_type*: string, optional
`-pin` *pin_name*	
	Specifies the name of a pin. The pin names are in the following format: *instance-name + default-hierarchyarchy-separator + pin*. This parameter supports wildcards (`*`) *Data_type*: string, optional
`-port` *port_name*	
	Specifies the name of a top-level port. The port names are in the following format: *top-level-port*. This parameter supports wildcards (`*`) *Data_type*: string, optional
`-quiet`	Suppresses all error and warning messages generated when the `set_switching_activity` command is run. *Data_type*: bool, optional
`-reset`	Resets all specified options back to default values. The reset option should be the first option specified, otherwise all options set prior to this will be reset. **Note:** You can selectively reset the pin/port/net specifications. In addition, you can also selectively reset the pin/port/net specifications of a list of objects through the `get_pins`/`get_ports`/`get_nets` commands. *Data_type*: bool, optional
`-scale` *value*	
	Specifies to scale frequencies of the specified clock domains. This parameter allows you to scale transition densities of all instances associated with the clock domain by the specified scale factor to compute internal and switching power. This parameter must be specified with the `-clock` parameter. The `-scale` parameter is only applicable to static power and dynamic vectorless power analysis. *Data_type*: string, optional
`-unclocked`	Applies *activity* (with reference *period*) to all nets that have no clock domain (`-unclocked`). To use this option, `-unclocked` must be used with `-activity` and `-period` with no other options. *Data_type*: bool, optional

Activity Precedence

See "Activity Precedence".

Examples

- The following command specifies that net `n189` will switch once every ten clock cycles, with a clock period of `7520ns` and a duty cycle of 50 percent:
  ```
  set_switching_activity -net n189 -activity 0.10 -period 7520ns -duty .5
  ```
- The following command sets the transition density of port `out_cpu_top` to `20000` transitions per second:
  ```
  set_switching_activity -port out_cpu_top -transition_density 20000
  ```
- The following command applies a transition density of `1e+06` to all input ports that match the specified pattern:
  ```
  set_switching_activity -port[get_ports "mode*" -filter {port_direction==in}] -transition_density 1e+06
  ```
- The following command resets the previously defined activity on all nets that match the specified pattern:
  ```
  set_switching_activity -reset -net[get_nets n23*]
  ```
- The following command specifies the clock gate ratio for the clock domain `sys_clk`:

```
set_switching_activity -clock sys_clk -integrated_clock_gate_ratio 0.8 -combinational_clock_gate_ratio 0.5
```

- The following command specifies to set the default switching activity for all other clock domains of the design:
  ```
  set_default_switching_activity -integrated_clock_gate_ratio 1.0 -combinational_clock_gate_ratio 1.0
  ```

Related Information

- `read_activity_file`
- `set_default_switching_activity`
- "Dynamic Power and IRDrop Analysis" in the *Voltus Stylus Common UI User Guide*

set_twf_attribute

```
set_twf_attribute
[-help]
[-clock clock_name]
[-clock_edge {rise | fall}]
[-clock_frequency value]
[-fall_delay {min:max or single_value}]
[-fall_slack {min:max or single_value}]
[-fall_slew {min:max or single_value}]
[-net net_name]
[-pin pin_name]
[-reset]
[-rise_delay {min:max or single_value}]
[-rise_slack {min:max or single_value}]
[-rise_slew {min:max or single_value}]
[-type {data | clock}]
[-shift_arrival_time time]
[-clear_twf_attribute]
```

Specifies static timing analysis information. The command allows you to override timing parameters, such as slew and clock, on desired pins/nets from an existing timing windows file (TWF).

Parameters

`-clear_twf_attribute`	
	Specifies to reset the TWF file values for a specific pin or net. The use model of this parameter is: `set_twf_attribute -clear_twf_attribute -pin/-net <name>` *Data_type*: bool, optional
`-clock clock_name`	
	Specifies the clock name. This indicates that the nets and pins that are timing constrained have arrival times (timing windows) caused by this clock. *Data_type*: string, optional
`-clock_edge {rise | fall}`	
	Specifies the capturing clock edge direction: rise or fall. *Data_type*: string, optional
`-clock_frequency value`	
	Specifies the clock frequency. You must specify the `-clock` parameter along with the `-clock_frequency` parameter to assign the clock for which the frequency needs to be updated. *Data_type*: string, optional
`-fall_delay {min:max` or `single_value}`	
	Specifies the fall delay for the pin. This parameter is not supported in the static power analysis flow. *Data_type*: string, optional
`-fall_slack {min:max or single_value}`	
	Specifies the fall slack for the pin. *Data_type*: string, optional

`-fall_slew {min:max or single_value}`	
	Specifies the fall slew for the pin. *Data_type*: string, optional
`-net net_name`	
	Specifies the net name. The `-net` and `-pin` parameters are mutually exclusive. *Data_type*: string, optional
`-pin pin_name`	
	Specifies the pin name. The `-net` and `-pin` parameters are mutually exclusive. *Data_type*: string, optional
`-reset`	Resets all specified options back to default values. *Data_type*: bool, optional
`-rise_delay {min:max or single_value}`	
	Specifies the rise delay for the pin. This parameter is not supported in the static power analysis flow. *Data_type*: string, optional
`-rise_slack {min:max or single_value}`	
	Specifies the rise slack for the pin. *Data_type*: string, optional
`-rise_slew {min:max or single_value}`	
	Specifies the rise slew for the pin. *Data_type*: string, optional
`-shift_arrival_time time`	

	Specifies to filter the idle period of the current waveform in the dynamic power analysis flow. This parameter is useful in removing the idle period of the current waveform result with long idle period (small current) before the 1st high current peak because of TWF delay annotation. The parameter allows to offset idle period from the TWF delay values.
	The use model is given below:
	`set_twf_attribute -shift_arrival_time 3.0ns`
	The default value is 0ns. The supported time units are s, ms, us, ns, and ps. The default unit is ns. You must specify a positive value only. A positive value means shift right in the time line or add more delay to the original arrival time.
	You can specify the `-clock` parameter to limit the changes to a clock domain. In addition, you can further add the `-pin/-net` parameter to limit the changes to a block. The following are examples of shifting different delays to different clock domains because different clocks may have different clock insertion delays at the top level:
	`set_twf_attribute -shift_arrival_time 1.2ns -clock clk_a`
	`set_twf_attribute -shift_arrival_time 2.3ns -clock clk_b`
	If a clock domain is not specified, the delay will be applied to all the specified nets/pins.
	Note: The final rise/fall delays for the annotated nets/pins after the shift has been applied are reported under the "Shift Arrival Time Report" section of the detailed TWF Attributes report. This report is generated using the `power_report_twf_attributes detailed` attribute.
	The following example illustrates a sample command and report snippet:
	Command:
	`set_twf_attribute -shift_arrival_time 3.0ns`
	Report:
	`Clock domain annotated`
	`  - CK1_1`
	`  - CK1_2`
	`  - CK2_1`
	`  - CK2_2`
	`Rise Delay  Fall Delay  Net Name`
	`1e-09  1e-09  CK1_1`
	`1e-09  1e-09  CK1_2`
	`1e-09  1e-09  CK2_1`
	`...`
	Data_type: string, optional

`-type {data | clock}`	
	Specifies whether the net or pin is part of the clock or data network. *Data_type*: string, optional

Examples

- The following command specifies to use the following TWF attributes for the net `netB1`:

  ```
  set_twf_attribute -net netB1 -rise_delay 2.1978:2.1978 -rise_slew 0.05:0.05 -rise_slack 0.2741:3.6961
  -fall_delay 2.1819:2.1819 -fall_slew 0.05:0.05 -fall_slack 0.237:3.6948 -clock CK1_1 -clock_frequency 100 -type data
  ```

- The following commands reset the TWF values for the "`DFF11/DFF/Q`" pin to default, and then sets the rise/fall delay:

  ```
  set_twf_attribute -clear_twf_attribute -pin "DFF11/DFF/Q"
  set_twf_attribute -pin "DFF11/DFF/Q" -rise_delay 0.35 -fall_delay 0.45 -clock CLK1
  ```

Related Information

- power Category Attributes
- `read_twf`
- `report_power`
- "Static Power, IRdrop and EM Analysis" in the *Voltus Stylus Common UI User Guide*
- "Dynamic Power and IRDrop Analysis" in the *Voltus Stylus Common UI User Guide*

set_virtual_clock_network_parameters

```
set_virtual_clock_network_parameters
[-help]
[-cell cell_name]
[-clock list_of_clocks]
[-library library_name]
[-max_fanout value]
[-wire_load_model wireload_model]
[-reset]
```

Specifies to estimate power for a virtual clock tree. When the command is specified, the software reports the power of the clock network based on the buffers inserted in the clock tree. It also reports the number of buffers used for implementing the clock tree and the depth of the clock tree.

You can use this command for fast power estimation of clock network power of a pre-CTS netlist.

This command should be invoked before running the `report_power` command.

Parameters

`-help`	Outputs a brief description that includes the type and default information for `each` set_virtual_clock_network_parameters parameter. For a detailed description of the command and all of its parameters, use the man command man `set_virtual_clock_network_parameters`.
`-cell cell_name`	
	Specifies the name of the buffer or inverter cell that has to be used to build the clock tree. *Data_type*: string, optional
`-clock list_of_clocks`	
	Specifies the list of clocks for which the clock tree needs to be implemented. The default is for all the clocks. *Data_type*: string, optional
`-library library_name`	
	Specifies the name of the library which contains the cell. *Data_type*: string, optional
`-max_fanout value`	
	Specifies the maximum fanout that the buffer cell can drive. *Data_type*: int, optional
`-wire_load_model wireload_model`	
	Specifies the wire load model used to compute the wire load model for the nets in the clock network. *Data_type*: string, optional
`-reset`	Resets all specified parameters back to default values. *Data_type*: bool, optional

Example

The following command specifies to use the cell `BUFX12`, with a maximum fanout value of `2`, to build a virtual clock tree:

```
set_virtual_clock_network_parameters -cell { BUFX12 } -library typical -max_fanout 2 -wire_load_model B0X0
```

Related Information

- `report_power`
- power Category Attributes
- "Static Power, IRdrop and EM Analysis" in the *Voltus Stylus Common UI User Guide*
- "Dynamic Power and IRDrop Analysis" in the *Voltus Stylus Common UI User Guide*

write_power_constraints

```
write_power_constraints
[-help]
[-out_file filename]
```

Writes out all the power constraints such as the power analysis mode, activity file specifications, simulation period specifications, and so on, into a file. You can use this command at any time during the session to dump out the power commands/ options specified till that point.

You can source this file into other Voltus/Tempus/Innovus session in order to use the same settings.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `write_power_constraints` parameter. For a detailed description of the command and all of its parameters, use the man command `man write_power_constraints`.
`-out_file filename`	Specifies the output file into which the power constraints are to be written. *Data_type*: string, optional

Example

```
write_power_constraints -out_file test1.tcl
```

Related Information

- power Category Attributes
- `report_power`
- "Static Power, IRdrop and EM Analysis" in the *Voltus Stylus Common UI User Guide*
- "Dynamic Power and IRDrop Analysis" in the *Voltus Stylus Common UI User Guide*

write_tcf

```
write_tcf file
[-block block_name]
[-exclude_hier_insts list_of_instances]
[-primary_input]
[-sequential_output]
[-pin]
[-macro_output]
```

Writes out the propagated switching activity information to a toggle count format (TCF) file. The TCF file contains switching activity information, toggle count information, and the probability of the net or pin being in the logic1 state.

You must specify this command before `report_power`.

Parameters

`-block block_name`	
	Specifies to generate the block-level TCF from the top-level design. When specified, it writes out the block-level duty and activity in the generated TCF file. The use model of this parameter is: `write_tcf -pin -block blockname TCF_filename`
`-exclude_hier_insts list_of_instances`	
	Excludes the specified hierarchical modules or instances when writing out a pin-based TCF. This parameter allows you to ignore the activity generated for the excluded instances. The use model of this parameter is: `write_tcf -pin -exclude_hier_insts {list_of_instances}`
`-macro_output`	Specifies to generate a TCF file that includes all the activities for the macro outputs in the design. *Data_type*: bool, optional
`-pin`	Specifies to write out pin-based TCF. *Data_type*: bool, optional
`-primary_input`	Generates a TCF file that includes only the primary inputs in the design. *Data_type*: bool, optional
`-sequential_output`	
	Generates a TCF file that includes all the activities for the sequential outputs in the design. *Data_type*: bool, optional
file	Specifies the name of the output TCF file. *Data_type*: string, optional

Example

- The following command writes the propagated switching activity information to TCF file `mmm.tcf`:
  ```
  write_tcf mmm.tcf
  ```
- The following command writes a pin-based TCF for the whole design:
  ```
  write_tcf -pin pin_design.tcf
  ```
- The following command writes out a TCF which only includes activity information for the primary inputs and sequential outputs in the design:

```
write_tcf -primary_input -sequential_output Power.tcf
```

- The following command writes out a pin-based TCF that excludes the specified hierarchical instances:
```
write_tcf -pin -exclude_hier_insts {DTMF_INST/CORE_INST DTMF_INST/RAM_128x16 DTMF_INST/RESULTS_INST
```

Related Information

- power Category Attributes
- `report_power`
- "Static Power, IRdrop and EM Analysis" in the *Voltus Stylus Common UI User Guide*
- "Dynamic Power and IRDrop Analysis" in the *Voltus Stylus Common UI User Guide*

27

Power-Grid Library Commands

- check_pg_library
- check_pgv
- merge_pg_library
- rename_pg_library
- set_advanced_pg_library_mode
- set_pg_library_mode
- trim_pg_library
- write_pg_library
- write_tech_pg_lib
- write_xpgv

check_pg_library

```
check_pg_library
[-help]
power_library_paths
[-cell_list_file file]
[-check_compatibility | -lef_consistency]
[-check_parameters]
[-out_dir file]
[-integrity_check]
[-list]
[-report]
[-rule_file file]
[-skip_layer_list_file filename]
[-skip_signal_net_check {true | false}]
[-summary]
[-tech_layers]
[-total_currents]
[-report_detail_multiple_voltage_cap file]
[-command_write]
[-metal_density_map]
```

Specifies to perform a comprehensive check of power-grid views (PGV) to be used in rail analysis. These checks include:

- Verify that all the PGV views exist: Early/IR/EM
- Report the dynamic PWL and static current
- Perform a compatibility check on version, layermap and resistivity parameters
- Check any missing information, such as temperature, qrcTechFile location, and so on.

Parameters

`-help`	Outputs the command usage.
power_library_path	

	Specifies the absolute path to the power library (`.cl`) file. You can run multiple power library files (.cl) at the same time. *Data_type*: string, required
`-cell_list_file` *file*	

	Specifies a file containing the list of cells for which the LEF consistency checks will be done. This parameter allows you to perform the consistency checks only for the user-defined cells.
	The format of the input file is:
	`CELL1`
	`CELL2`
	`CELL3`
	Data_type: string, optional

`-check_compatibility`	
	Checks for compatibility between the specified cell libraries. It checks whether the following parameters are the same for the libraries that are to be merged: • Time Units • Capacitance Units • Layout Scale Factor • Manufacturing Grid If one or more of these parameters are not the same, a warning message appears stating that the libraries are not compatible and cannot be merged. It also specifies the parameters that are different. Based on the above electrical/physical parameter check, a compatibility report (compatibility.report) with pass/fail is generated. Fail implies that the library is not compatible and cannot be used in Voltus rail analysis. The use model of this parameter is: `check_pg_library -check_compatibility {`*lib1 lib2*`}` *Data_type*: bool, optional
`-check_parameters`	

	Checks for the following electrical parameters of the specified cell libraries: • `CINT_MIN and CINT_MAX`: Intrinsic device capacitance • `CG_MIN and CG_MAX`: Grid capacitance • `RON_MIN and RON_MAX`: On-resistance for powergates • `IDSAT_MIN and IDSAT_MAX`: Saturation current for powergates • `ILEAK_MIN and ILEAK_MAX`: Leakage current for powergates • `IAVG_MIN and IAVG_MAX`: Static current • `IPEAK_MIN and IPEAK_MAX`: Dynamic peak current • `ESR_MIN and ESR_MAX`: Equivalent series resistance For powergates, `-check_parameters` will check the Vds-Ids table for the correct trend. It checks whether: • for a given `Vgs`, `Ids` increases with `Vds` • for a given `Vds`, `Ids` increases with `Vgs` where; `Vgs` is voltage across gate and source `Vds` is voltage across drain and source `Ids` is current from drain to source (of a transistor) *Data_type*: bool, optional
`-command_write`	
	Specifies to print the settings used during PGV characterization to a text file called `command_list.txt`. This parameter allows you to determine the user-specified or default settings used during PGV generation for the specified power library (.cl). *Data_type*: bool, optional
`-integrity_check`	
	Specifies to load all the views from a power-grid library to ensure that the library contains the required data. This parameter checks if there are any missing files or views, and that the PGV is usable in rail analysis.
`-lef_consistency`	
	Checks for LEF consistency of the cells in the specified cell libraries. It reads the LEF and PGV files, and compares the given LEF file with the LEF file used in PGV generation. *Data_type*: bool, optional

`-list`	Specifies to list the PGVs and the cells in each PGV. The parameter allows you to determine the name of the PGV library that contains a specific cell and determine whether a cell is there in multiple libraries.
	
	The format of the output file is:
	`PGV Library: <Number of PGV libraries>`
	`<Lib_1> <TOOL VERSION> <PGV VERSION> <GENERATE_DATE> <Path>`
	`<Lib_2> <TOOL VERSION> <PGV VERSION> <GENERATE_DATE> <Path>`
	`...`
	`Duplicated Cells: <Number of cells in multiple libraries>`
	`<CELLNAME> <Lib_1 Lib_2 Lib_3>`
	`Cell List: <Number of cells in a library>`
	`<CELLNAME> <LIB#>`
	
	Data_type: bool, optional

`-metal_density_map`	
	Generates the metal density map report file (`<pgvname>.metal_density`) in the PGV output directory. This file contains information about the macro size, number of tiles in the x and y direction, metal layer ID, layer name, and metal density values for each tile. This parameter is used with the `-calculate_metal_layer_density` parameter of the `set_pg_library_mode` command. *Data_type*: bool, optional
`-out_dir dir_name`	
	Specifies the name of the output directory in which the power-grid library report is created. The default is the current working directory. *Data_type*: string, optional
`-report`	Specifies to generate the power-grid library report. *Data_type*: bool, optional
`-report_detail_multiple_voltage_cap file`	
	Specifies to generate a voltage-dependent capacitance table that contains the detailed capacitance values of the given power-grid library. The capacitance table file is called `<pgvname>.cap_table`. This parameter is used with the `-enable_multi_voltage_cap_generation` parameter of the `set_pg_library_mode` command. *Data_type*: bool, optional
`-rule_file file`	

	Specifies the rule file name used for parameter checking. This file contains the list of electrical parameters to be checked.
	File Format:
	`CINT_MIN <min cap value>`
	`CINT_MAX <max cap value>`
	`RON_MIN <value>`
	`RON_MAX <value>`
	`IDSAT_MIN <value>`
	`IDSAT_MAX <value>`
	`ILEAK_MIN <value>`
	`ILEAK_MAX <value>`
	`IPEAK_MIN <value>`
	`IPEAK_MAX <value>`
	`IAVG_MIN <value>`
	`IAVG_MAX <value>`
	`ESR_MIN <value>`
	`ESR_MAX <value>`
	This parameter must be specified with the `-check_parameters` parameter.
	Data_type: string, optional

`-skip_layer_list_file` *filename*	
	Specifies the name of a file containing a list of layers to ignore while checking PGV and LEF compatibility. Its format is one layer per line. This parameter must be specified with the `-check_compatibility` parameter. *Data_type*: string, optional
`-skip_signal_net_check {true | false}`	
	Default: `false` Specifies to skip LEF consistency checks for signal nets. This forces skipping of the signal pin checks when performing a comprehensive check of power-grid views to be used in rail analysis. The `-skip_signal_net_check` parameter can be used if you want to check LEF consistency of only the power and ground pins. *Data_type*: string, optional
`-summary`	
	Specifies to generate a detailed report of the library contents for the specified cells that can be used for debugging purpose. *Data_type*: bool, optional
`-tech_layers`	
	Specifies to report technology layer information and layer mapping. *Data_type*: bool, optional
`-total_currents`	
	Specifies to report the static current for each power/ground net, and dynamic PWL for each mode. For each mode, it reports the peak and average current, and the number of points for power supply nets. When specified, the total current is reported in `<lib_name>.total_current` for each library. *Data_type*: bool, optional

Examples

- The following command creates a report for the library std_cells.cl:
  ```
  check_pg_library std_cells_fast/std_cells_fast.cl -report
  ```
- The following command checks for LEF consistency:
  ```
  read_lib -lef ddk.0.lef
  read_lib -lef ddk.1.lef
  check_pg_library -lef_consistency \
  "/../library1.cl\
  /../library2.cl"
  ```
- The following command displays the list of layers (defined in the technology file) on stdout:
  ```
  check_pg_library -tech_layers ./check_param/powergate/stdcells.cl
  ```
- The following command checks for compatibility between the specified cell libraries and generates a compatibility report (compatibility.report) in the output directory:
  ```
  check_pg_library -out_dir out -check_compatibility  {/../lib1.cl /../lib2.cl}
  ```
- The following command specifies to list the cells corresponding to the specified libraries, `pgv1.cl` and `pgv2.cl`:
  ```
  check_pg_library -list {/../pgv1.cl  /../pgv2.cl}
  ```
- The following command checks for electrical parameters of the stdcells.cl library:
  ```
  check_pg_library -out_dir OUT -check_parameters ./stdcells.cl
  ```
  The following is a sample output of this command:

  ```
            PARAM        VALUE            RANGE         PASS/FAIL

  Cell: Cell1
     Net VDDG
          Device Cap    4.50e-14    (1.00e-18,1.00e-11)    PASS

     Net VSS
          Device Cap    2.61e-14    (1.00e-18,1.00e-11)    PASS

     Net VDD
          Device Cap    2.13e-14    (1.00e-18,1.00e-11)    PASS

     Power Gate SLEEP
               Ron      1.84e+01    (2.00e+01,1.00e+03)    FAIL
               Ileak    7.01e-08    (1.00e-15,1.00e-06)    PASS
               Idsat    3.36e-02    (1.00e-09,1.00e-02)    FAIL

     Trend Checks:
         Ron Trend   : PASS
         Idsat Trend  : PASS
         Ileak Trend : PASS
  ```
- The following command reports the static current for each power/ground net:
  ```
  check_pg_library -total_currents /../pg.cl
  ```
  The following is a sample output of this command:

  ```
  Net VSS
  Static current = -0.00021181

  Net VDD
  Static current = 0.000211809

  MODE1
  Trigger Condition: ( !CEN & !WEN )
  Net VDD
      Peak = 0.0138308
      Avg = 0.000276053
      Number of points 190

  Net VSS
      Peak = -0.0166879
      Avg = -0.000300184
      Number of points 190
  ```

```
MODE2
...
```

- The following command generates a capacitance table (`powergate_stdcells.cap_table`) that contains the detailed capacitance values of the powergate_stdcells.cl library:

  ```
  check_pg_library -report_detail_multiple_voltage_cap ../powergate_stdcells.cl
  ```

  The following is a snippet of the capacitance table file (`powergate_stdcells.cap_table`):

  ```
  ================================================================================
  CellName    NetName      Voltage       Capacitance
  --------------------------------------------------------------------------------
  powercell    vccm        0.1000        4.586e-14
  powercell    vccm        0.2825        4.102e-14
  ...
  ```

- The following command specifies to output the settings used during generation of `macros_pll.cl` to a text file called `command_list.txt`:

  ```
  check_pg_library -command_write macro_pgv/macros_pll.cl
  ```

  The following is a sample snippet of the `command_list.txt` file:

  ```
  temperature 25
  spice_subckts /../spice.sp
  spice_models /../spectre_load.sp
  extraction_tech_file /../tech.tch
  gds_files /../file1.gds
  stop@via VIA_1
  spice_subckts_xyscaling 1000
  use_powergate_data_from_lib 0
  default_frequency 2e+08
  decap_frequency 1e+08
  ```

- The following command generates the power-grid library report:

  ```
  check_pg_library -out_dir report -report /../pg.cl
  ```

  The following is a sample output of this command:

  ```
  ================================================================================

  Library path name:        /it/../library
  Library name:            pg
  Hierarchical library:    NO
  User login:              user
  Date:                    2015-Sep-03 03:58:45 (2015-Sep-03 10:58:45 GMT)
  ToolIdentifier:          VOLTUS
  Application version:     15.20-d127_1
  Number of cells in library:    1

  ================================================================================
  Cell                PowerNets       Capacitance      PowerGrid Views
  ------------------------------------------------------------------------------------------------------------------------|

  cell1

                      VSS(0.0000)     5.3e-12          EARLY(255 taps) EM(8827 taps) IR(8827 taps)
                      VDD(0.9000)     6.8e-12          EARLY(193 taps) EM(7534 taps) IR(7534 taps)

   Cell Type = MACRO                                    Cell_Status: PASS

  INFO: Dynamic Tap Current views were created for this cell. MODE1 has Conditional input = "( !CEN & !WEN ) " and Conditional pin = "CLK
  RISE".
  INFO: Dynamic Tap Current views were created for this cell. MODE2 has Conditional input = "( CEN & WEN ) " and Conditional pin = "CLK
  RISE".

  ------------------------------------------------------------------------------------------------------------------------|
  ```

Related Information

- check_pgv
- "Power-Grid Library Generation" in the *Voltus Stylus Common UI User Guide*

check_pgv

```
check_pgv
[-help]
[-activity value ]
-cell cell_name
[-clock {clk1 clk2 ...}]
[-disable_rail_analysis {true | false}]
[-dynamic_analysis_duration value]
[-enable_xp {true | false}]
-extraction_tech_file tech_file_path
[-frequency value]
[-input_pin_set_case_analysis_low {pin1 pin2 ...}]
[-input_pin_set_case_analysis_high {pin1 pin2 ...}]
[-liberty liberty_file ]
[-max_ir_drop value]
[-net_capacitance value]
[-output output_dir]
[-pin_capacitance value]
-power_grid_library_path pgv_path
[-power_grid_library_view view_name]
[-power_pad_size value]
[-rail_analysis_type {static | dynamic} ]
[-sdc_file filename]
[-set_power_switching_pattern trigger_file]
[-set_total_power value]
[-step_size value]
[-supply_voltage value]
-tech_lef_file tech_lef_file_path
[-transition_time time]
[-power_include_file filename]
[-rail_include_file_begin filename]
[-rail_include_file_end filename]
[-rail_trigger_switching_pattern pattern]
[-temperature value]
[-twf_path twf_location]
[-user_defined_power_pin_location_file filename]
[-pgv_mode_dynamic_switch_pattern pattern]
```

Specifies to validate a macro PGV. The command takes a macro PGV and Liberty file as input, and performs IR Drop based validation by running static and dynamic power and rail analysis. In this flow, Library Generation will read the input PGV and generate a dummy DEF file that has an instance of the macro (one-cell design). It will then run static/dynamic power and rail analysis on this design. You can analyze the power and rail results to check for hotspots, connectivity, and current distribution inside the PGV.

Parameters

`-help`	
	Outputs the command usage.
`-activity value`	
	Specifies the activity value for input pins. If not specified, a default activity of 0.2 is used. The activity specifies the number of times an input pin switches *Data_type*: string, optional
`-cell cell_name`	
	Specifies the name of the cell to be validated. *Data_type*: string, required
`-clock {clk1 clk2 ...}`	
	Specifies the list of clock pins. *Data_type*: string, optional

`-disable_rail_analysis {true | false}`

Specifies to disable rail analysis during macro PGV validation. The following is the behavior of `-disable_rail_analysis true` with the `-enable_xp` para

- if `-enable_xp` is set to `true`, the tool will run only power analysis in the XP mode
- if `-enable_xp` is set to `false`, the tool will enable only power analysis in the non-XP or normal mode

Default: `false`

Data_type: enum, optional

`-dynamic_analysis_duration value`

Specifies the simulation period that dynamic power analysis will use. It is the duration for which a window is selected for calculation of IR drop during pc default value is 10ns.

Data_type: string, optional

`-enable_xp {true | false}`

Specifies to validate the macro PGV in the extensively parallel (XP) mode.

Default: `false`

`-extraction_tech_file tech_file_path`

Specifies the path of the extraction technology file.

Data_type: string, optional

`-frequency value`

Specifies the clock frequency. The default value is 100Mhz.

Data_type: string, optional

`-input_pin_set_case_analysis_low {pin1 pin2 ...}`

Specifies the list of pins to be kept at a logical low. The `set_case_analysis` command in the SDC file will have the same effect as the set pin to constant described here.

Data_type: string, optional

`-input_pin_set_case_analysis_high {pin1 pin2 ...}`

Specifies the list of pins to be kept at a logical high. The `set_case_analysis` command in the SDC file will have the same effect as the set pin to constan described here.

Data_type: string, optional

`-liberty liberty_file`

Specifies the name of the Liberty file for power calculation.

This parameter is optional for the static power analysis flow, and is mandatory for the dynamic power analysis flow. In the static flow, you need to specif `set_total_power`. If both the parameters are specified, the `-set_total_power` parameter will be honored.

Data_type: string, optional

`-max_ir_drop value`

Specifies the maximum IR drop allowed. Default is 5 (in percentage).

Data_type: float, optional

`-net_capacitance value`

Specifies the value of the net capacitance. The default value is 1pF.

Data_type: string, optional

`-output output_dir`

Specifies the name of the output directory. This directory saves the power and rail reports, and the design output (such as .lef, .def, and so on).

Data_type: string, optional

`-pgv_mode_dynamic_switch_pattern` *pattern*

Specifies the switching patterns for the functional modes in the dynamic power calculation flow. The use model of this parameter is same as the `set_pow` `dynamic_switch_pattern` parameter.

In the following example, the mode `write` will switch in the 1st clock cycle, the mode `read` will switch in the 2nd and 4th clock cycles, and none of the mo 3rd clock cycle:

```
check_pgv -pgv_mode_dynamic_switch_pattern { write read - read }
```

`-pin_capacitance` *value*

Specifies the value of the pin capacitance. The default value is 1pF.

Data_type: string, optional

`-power_grid_library_path` *pgv_path*

Specifies the path of the power-grid library file.

Data_type: string, required

`-power_grid_library_view` *view_name*

Specifies the power-grid view name.

Data_type: string, optional

`-power_include_file` *filename*

Specifies a power analysis include file. This file specifies certain commands of the dynamic engine that do not have a Voltus equivalent. This parameter the dynamic power analysis flow.

Data_type: string, optional

`-power_pad_size` *value*

Specifies the distance between two adjacent DEF voltage sources. The default value is `50` (in Micron).

Data_type: float, optional

`-rail_analysis_type {static | dynamic}`

Specifies whether static or dynamic analysis will be done. The default value is `static`.

Data_type: string, optional

`-rail_include_file_begin` *filename*

Provides a file that includes additional rail options that will be run prior to the `analyze` command. This can be used for those legacy rail analysis comma Voltus equivalent.

Data_type: string, optional

`-rail_include_file_end` *filename*

Provides a file that includes additional rail options that will be run after the `analyze` command. This can be used for those legacy rail analysis command Voltus equivalent.

Data_type: string, optional

`-rail_trigger_switching_pattern` *pattern*

Default: None

Specifies a dynamic trigger file. When a Macro EM view with current signature is used in a design, it needs to be determined when this block/cell is trigg trigger file is used to determine this. The use model of this parameter is same as the `set_rail_analysis_config -dynamic_trigger_file` parameter.

Data_type: string, optional

`-sdc_file` *filename*

Specifies the name of the timing constraints file.

Data_type: string, optional

`-set_power_switching_pattern` *trigger_file*

Default: None

Specifies the stimulus pattern on the pins of an instance and cell. Specify this parameter only in the dynamic mode for the type instance and cell. The us
parameter is same as the `set_power -dynamic_switch_pattern` parameter.

Data_type: string, optional

`-set_total_power value`

Specifies the total power for the design in Watts. This power is distributed based on cell area.

Data_type: string, optional

`-step_size value`

Specifies the simulation step size, in picoseconds. The default value is 20ps. This parameter is applicable only to the dynamic power analysis flow.

Data_type: string, optional

`-supply_voltage value`

Specifies the voltage of each power supply pin. The default value is 1V.

Data_type: float, optional

`-tech_lef_file tech_lef_file_path`

Specifies the path of the technology LEF file.

Data_type: string, optional

`-temperature value`

Sets the temperature for static power calculation. The default is 125 degrees.

Data_type: string, optional

`-transition_time time`

Specifies the transition time for specific pins or instances in the design. This parameter is applicable only to the dynamic power analysis flow.

Data_type: string, optional

`-twf_path twf_location`

Specifies the location of the Timing Window File (TWF) for noise and power calculation.

Data_type: string, optional

`-user_defined_power_pin_location_file filename`

Specifies the name of a file containing net names and corresponding user-defined ploc or power pad files (.pploc). The .pploc file includes power pin or
locations for power and ground nets. The parameter gives you the flexibility to include either user-specified or tool-generated voltage sources.

The following is a sample file with the list of nets and power pad files:

```
----ploc.list----
#<NET_NAME> <Pointer to ploc file>

VDD   input/ploc/VDD.pploc
VDDA  input/ploc/VDDA.pploc
VDDF  input/ploc/VDDF.pploc
VSS   input/ploc/VSS.pploc
```

When specified, the voltage source location files (.pploc) in the given *`ploc.list`* file will be used for library generation. If the pploc file is missing for a fe
file, then library generation will automatically generate `.pploc` for those nets.

Data_type: string, optional

Examples

- The following command specifies to validate the cell `cell1` by running static power and rail analysis:
  ```
  check_pgv -cell cell1 -liberty 125c.lib -clock {CLKW CLKR} -power_grid_library_path macros_cell.cl -extraction_tech_file caps.tch -tech_lef_file tech.lef -output validate_pg_output_static -rail_analysis_type static
  ```
- The following command specifies to validate the cell `cell1` by running dynamic power and rail analysis
  ```
  check_pgv -cell cell1 -liberty 125c.lib -clock {CLKW CLKR} -power_grid_library_path macros_cell.cl -temperature 122 -extraction_tech_file caps.tch -tech_lef_file tech.lef -output validate_pg_output_dynamic -rail_analysis_type dynamic
  ```

Related Information

- `check_pg_library`
- "Power-Grid Library Generation" in the *Voltus Stylus Common UI User Guide*

merge_pg_library

```
merge_pg_library
[-help]
[-force_merge {true | false}]
-library_list_file file_name
[-library_prefix prefix]
[-merge_single_voltage_pgvs {true |false}]
[-out_dir directory_name]
[-remove_tech {true | false}]
[-validate_rail_merge {true | false}]
```

Specifies to merge power-grid libraries. Using this command, you can merge multiple same-type power-grid libraries into one consolidated power-grid library.

Parameters

`-help`	Outputs the command usage.
`-force_merge {true | false}`	
	Specifies to force merge PGVs with different resistivity and layers. It uses the technology information of the first library. If multiple tech-only PGVs are specified, it will take the first tech-only PGV definition and the first cell definition. If you are merging a tech-only PGV with other PGVs, it will keep the tech-only technology definition and the first cell definition. The default value of the parameter is `false`. *Data_type*: string, optional
`-library_list_file` *file_name*	
	Specifies the file containing the list of libraries. *Data_type*: string, required
`-library_prefix` *prefix*	
	Specifies a prefix for the generated power-grid library. *Data_type*: string, optional
`-merge_single_voltage_pgvs {true |false}`	
	Specifies to merge all the single voltage PGVs into a multi-voltage PGV for better handling of the library characterization flow. This parameter is used in a scenario when you have generated multiple PGVs for different voltages, instead of generating a single multi-voltage PGV that has voltage-dependent capacitance. The default value of the parameter is `false`. *Data_type*: string, optional
`-out_dir` *directory_name*	
	Specifies the name of the output directory in which power-grid libraries are created. The default is the current working directory. **Note**: If you specify both `-out_dir` *output_dir* and `-library_prefix` *prefix* parameters, the command generates the PGV in the *output_dir* directory with PGV name `prefix.cl`. *Data_type*: string, optional
`-remove_tech {true | false}`	
	Specifies to remove technology dependency from the specified or merged library. The newly created library will not contain the technology information. The default value of the parameter is `false`. *Data_type*: string, optional
`-validate_rail_merge {true | false}`	
	Specifies to validate the merged PGV in the same way as done when merging PGV during rail analysis. When this parameter is specified, the log file reports error, warning, and info messages if the tool finds any issues during PGV merging. The parameter allows you to check the PGV merging issues as reported during rail analysis, without having to run the full design. `-validate_rail_merge` is used only for validation purpose, and the merged PGV is not to be used for analysis. *Data_type*: string, optional

Examples

- The following command merges the power-grid libraries specified in the `merge_lib.list` file into one consolidated power-grid library:
  ```
  merge_pg_library -library_list_file "merge_lib.list" -out_dir ./ -library_prefix "new_merged"
  ```
  The content of `merge_lib.list` file is:
  library1.cl
  library2.cl
  library3.cl

Related Information

- `write_pg_library`
- "Power-Grid Library Generation" in the *Voltus Stylus Common UI User Guide*

rename_pg_library

```
rename_pg_library
[-help]
-cell_list_map_file filename
-merge_library_list_file filename
-out_dir directory_name
-output_library new_library_name
```

Specifies to copy and rename the specified power-grid view library. The command also allows you to rename the cells of the specified input library.

Parameters

`-help`	Outputs the command usage.
`-cell_list_map_file` *filename*	
	Specifies a file containing the list of cells to be renamed. The format of the specified file is: *<old cell name1> <new cell name1>* *<old cell name2> <new cell name2>* ... *Data_type*: string, required
`-merge_library_list_file` *filename*	
	Specifies a file containing the input library to be renamed. The format of the specified file is: *<path to the input library>* *Data_type*: string, required
`-out_dir` *directory_name*	
	Specifies the name of the output directory in which the new power-grid library is created. The default is the current working directory. *Data_type*: string, required
`-output_library` *new_library_name*	
	Specifies the new name of the specified input library. *Data_type*: string, required

Examples

- The following command specifies to rename the input library to `new_lib`:

```
rename_pg_library -merge_library_list_file new.list -cell_list_map_file cell.list -output_library new_lib -out_dir .
```

Related Information

- `set_pg_library_mode`
- `set_advanced_pg_library_mode`
- `write_pg_library`
- "Power-Grid Library Generation" in the *Voltus Stylus Common UI User Guide*

set_advanced_pg_library_mode

```
set_advanced_pg_library_mode
[-add_port_labels file_name]
[-assume_foreign {true | false}]
[-assume_foreign_mode [1 | 0]]
[-cell_electrical_data_file file]
[-cell_pin_net_map_file file]
[-circuit_include_file thunder.inc]
[-cluster_via_rule {{via_layer1 number_of_equidistant_vias}...}]
[-cluster_via_size value]
[-common_supply_pins {net_name+}]
[-create_static_view_from_dynamic_view {true | false}]
[-damping_decap_cells {cell1 cell2 ..}]
[-damping_decap_frequency value]
[-default_frequency value]
[-default_power_voltage value]
[-disable_power_switch_ramp_up_simulation {true | false}]
[-esd_cells {cell_list}]
[-extraction_command_file file]
[-import_xdspf_list_file filename]
[-pg_library_generation_command_file file]
[-macro_parasitic_file filename]
[-marker_layer_map filename]
[-pgdb_list_file filename]
[-pgdb_layer_map filename]
[-power_switch_characterization_voltages { val1 val2 val3 .... }]
[-power_switch_multi_enable_characterization_method {enable_all | enable_single} ]
[-process_bulk_diffusion_ports {true|false}]
[-remove_em_view_dangling_resistor {true|false}]
[-schematic {true|false} ]
[-source_location_file {filename}]
[-techgen_dir directory]
[-thunder_command_file file]
[-verbosity {true | false}]
[-well_capacitance_file filename]
[-xdspf_layer_map filename]
[-xtc_command_file filename]
[-xtc_include_file filename]
[-followpins_tap_layer {lowest_lef_pin_layer | highest_lef_pin_layer | all_lef_pin_layers}]
[-followpins_interface_node_layer {lowest_lef_pin_layer | highest_lef_pin_layer | all_lef_pin_layers}]
[-use_embedded_spectre {true | false}]
[-strict_input_check {true|false}]
[-ignore_pg_nets {{cellname netname}+}]
[-tap_node_distance value]
[-decap_frequency value ]
[-input_port_value_list {pin1 value1 pin2 value2 ....pinN valueN}]
[-exclude_tap_region_file filename]
[-skip_switch_net_extraction {true | false}]
[-delete_generated_fsdb_files {true|false} ]
[-lef_layer_ignore_list {lef_layername+}]
[-lef_layer_ignore_list_file filename]
[-quit_on_extraction_errors {true|false}]
[-esd_cells_list_file filename]
[-generate_interface_node_x_direction {true | false}]
[-generate_interface_node_y_direction {true | false}]
[-on_resistance_measure_threshold value]
[-cell_per_distributed_host value]
[-distribute_current_to_switch_net {true | false}]
[-esd_pin_list {pin1 pin2 ....pinN}]
[-esd_device_list {device1 device2 ....deviceN}]
[-generate_interface_nodes_at_via_layer {true | false}]
[-enable_subconductor_layers {true | false}]
[-enable_ac_simulation_for_IO_characterization {true | false}]
[-esd_parameters_file filename]
[-generate_bulk_pin_cap_separately {true | false}]
[-stdcell_characterization_voltage_value {val1 val2 val3 ...}]
```

```
[-spectre_path value]
[-enable_spectre_netlist_flow {true | false}]
[-interface_node_location_file filename]
[-generate_gray_box_data {true | false} ]
[-lef_pin_short {true | false}]
[-lef_pin_short_cell_file filename]
[-lef_pin_short_cell_list {cell1 cell2 ...}]
[-redo_characterization_count count]
[-retain_generated_pgv_on_error {true | false}]
[-trim_metal_layer_map file]
[-use_peak_current_distribution {true | false}]
[-enable_via_based_current_distribution {true | false}]
[-via_based_current_config_file filename]
```

Specifies the advanced power-grid library generation features. If the `set_advanced_pg_library_mode` command is specified twice, the last specified command takes precedence.

Parameters

`-add_port_labels file_name`

Specifies the name of a file containing details about the label of a switched rail. Using this parameter, you can add a label for the switched rail manually in the specified port file, if the switched rail is not labeled.
The format of the port file is given below:

MACRO *macro_name*
PIN *pin_name*

Data_type: string, optional

`-assume_foreign {true | false}`

Enables the Library Characterization engine to process LEF files in which FOREIGN statements are not defined as part of the macro definition. When generating detailed power-grid views, the engine assumes that FOREIGN statements are defined as part of the macro definition in the LEF file.

When this option is set to `true`, the engine processes the LEF file by using the name defined by the MACRO statement rather than requiring a FOREIGN statement. The `-assume_foreign` parameter ensures that the FOREIGN statement is placed at 0,0. Therefore, if the MACRO statement has an offset, you must provide a FOREIGN statement that contains an offset. This option works for offsets that are 0,0 or the inverse of the origin.

Default: false

Data_type: enum, optional

`-assume_foreign_mode [1 | 0]`

Determines how the Library Characterization engine processes cells for which FOREIGN statements are assumed. If set to 0, the engine assumes that the cell for which FOREIGN statements are assumed is placed at (0,0). When you set it to 1, the engine assumes that the FOREIGN cell is placed at the inverse of the macro origin. The default is 1.

Default: 1

Data_type: enum, optional

`-cell_electrical_data_file file`

Specifies the name of a file containing data about the electrical characteristics of cells. Voltus library simulator requires this data to generate currents during power-grid view generation.

Data_type: string, optional

`-cell_per_distributed_host value`

Specifies the number of cells to be submitted per host for distributed PGV generation. The default value is 50.

Data_type: int, optional

`-cell_pin_net_map_file file`

	Specifies the mapping between the cell LEF pin names and the internal names of the net to which those pins are attached. It is used when multiple pins are attached to a single net. *Data_type:* string, optional
`-circuit_include_file` *`simulator.inc`*	
	Specifies the name of the input file for Voltus library simulator in the Library Characterization command file. *Data_type:* string, optional
`-cluster_via_rule {{`*`via_layer1 number_of_equidistant_vias`*`}...}`	
	Controls the number of vias to cluster on a layer basis. The VIA clustering rule specified using this parameter will override the default clustering rule for PGV generation. You can use this parameter when you need clustering at VIA1, or different clustering for different layers. The default cluster size for PGV generation is `64` for all via layers except VIA1. **Example**: The following command will cluster 100 equidistant VIA1 cuts, 200 equidistant VIA2 cuts, and 300 equidistant VIA7 cuts: `set_advanced_pg_library_mode -cluster_via_rule {{VIA1 100} {VIA2 200} {VIA7 300}}` *Data_type:* string, optional
`-cluster_via_size` *`value`*	
	Specifies that power-grid extraction for library cell needs to perform 64 (8x8) via clustering inside PGV for all via layers above VIA1 between Metal1 and Metal2. The default value of this parameter is `64`. The following command specifies to perform 16 (4x4) via clustering: `set_advanced_pg_library_mode -cluster_via_size 16` *Data_type:* int, optional
`-common_supply_pins{`*`net_name+`*`}`	
	Identifies the generic supply net or nets which could act as either a power or ground supply net. No wildcards are allowed for the specification of net names. Generic supply nets are used for pads or bump cells that could be connected to either a power or ground net. They will be connected to the correct power-grid based on the DEF or GDS file. *Data_type:* string, optional
`-create_static_view_from_dynamic_view {true | false}`	
	Specifies to generate static power-grid view by averaging of dynamic currents in the multi-mode characterization flow. When this parameter is specified, the software performs averaging of the dynamic currents of all the modes and saves it as static current in the power-grid view. *Default*: `false` *Data_type:* enum, optional
`-damping_decap_cells {`*`cell1cell2 ..`*`}`	
	Specifies a list of damping decap cells. *Data_type:* string, optional
`-damping_decap_frequency` *`value`*	
	Specifies the frequency of operation of the damping decap cells. The default value is 200MHz. *Data_type:* float, optional
`-decap_frequency` *`value`*	

	Specifies the decoupling capacitance frequency (in hertz) used during PGV generation. Intrinsic capacitance of cells has a significant dependency on frequency used during PGV generation. PGVs characterized with higher frequency may lead to pessimism during dynamic analysis due to less impact of intrinsic capacitance of the cells characterized with higher frequency. Using this parameter, you can specify the dominant frequency to be used for PGV generation. The default value for decoupling capacitance frequency is 1GHz. For example, the following command specifies that the decap frequency is 200 MHz: `set_advanced_pg_library_mode -decap_frequency 2e+8` *Data_type:* string, optional
`-default_frequency` *value*	
	Specifies the default frequency of signal nets in hertz (Hz); this is used to calculate the current and capacitance values during the creation of detailed power-grid views. For example, the following command specifies that the default switching frequency is 100 MHz: `set_advanced_pg_library_mode -default_frequency 100e+06` *Data_type:* string, optional
`-default_power_voltage` *value*	
	Specifies the default voltage for power nets. This parameter assigns a default value to power nets for which voltage has not been defined using the `-power_pins` parameter. *Data_type:* float, optional
`-delete_generated_fsdb_files {true|false}`	
	Specifies to delete the intermediate FSDB files generated in the Spice deck (user-specified) method of current characterization. This parameter allows you to retain or delete the huge FSDB files. *Data_type:* enum, optional
`-disable_power_switch_ramp_up_simulation {true | false}`	
	Specifies whether to run power gate ramp-up simulation during power-grid view generation. When this parameter is set to `true`, the Ids-Vds table will not be present in PGV, therefore, this PGV cannot be used for the power-up or rush current analysis flow. This parameter can be specified to reduce the run time when you require only the RON/Idsat/Ileak and Cap values in PGV. `-disable_power_switch_ramp_up_simulation` is used when power gate parameters are specified using the `set_pg_library_mode -power_switch_parameters_file` parameter. *Default*: `false`
`-distribute_current_to_switch_net {true | false}`	
	Distributes the PWL waveform, which is specified to the always-on power net, to the taps of both the always-on power net and the related switched power nets in the fine-grain memory characterization flow. By default, the current waveform is distributed only to the taps of the always-on power net when you specify the PWL to the always-on power net. *Default*: `false` *Data_type:* enum, optional
`-enable_ac_simulation_for_IO_characterization {true | false}`	
	Specifies to perform an AC analysis based coupled capacitance calculation for IO cells. When set to `false`, the software performs DC analysis based coupled capacitance calculation for IO cells. The default is `false`. *Data_type:* enum, optional
`-enable_spectre_netlist_flow {true | false}`	
	Specifies to use the Spectre simulator to read the Spice/Spectre netlist for the standard cells characterization flow. This option should be used when the netlist is in Spectre format, or when the netlist is encrypted by Spectre. *Default*: `false` *Data_type:* enum, optional

`-enable_subconductor_layers {true | false}`

Allows to control the generation of the sub-conductor layers in the PGV. When this parameter is set to `false`, the sub-conductor layers are not generated in the PGV. The default value of this parameter is `true`.

Data_type: enum, optional

`-enable_via_based_current_distribution {true | false}`

Default: `false`

Specifies to enable location-based via current modelling during GDS-based macro power-grid view generation. When set to `true`, defines the current values only on the user-defined via location. This parameter allows you to define average and peak current values for specific-location vias.

The `-enable_via_based_current_distribution` parameter must be specified with the `-via_based_current_config_file` parameter.

If the parameter is set to `false`, current is uniformly distributed across all vias in the macro PGV.

Data_type: enum, optional

`-esd_cells {cell_list}`

Specifies a list of Electrostatic Discharge (ESD) cells. You can use this parameter to characterize ESD cells. When you specify this parameter, the list of ESD cells are later used in the ESD analysis flow while loading the power-grid view library.

Data_type: string, optional

`-esd_cells_list_file filename`

Specifies a file containing a list of ESD cells that are later used in the ESD analysis flow while loading the power-grid view library.

The `-esd_cells_list_file` parameter is the same as the existing `-esd_cells` parameter of the `set_advanced_pg_library_mode` command, except that the `-esd_cells_list_file` parameter allows you to specify a list of ESD cells in a file while the `-esd_cells` parameter allows you to specify the ESD cells in the command line itself.

The following is an example of this parameter:

```
set_advanced_pg_library_mode -esd_cells_list_file {./Esd_Cell_list.txt}
```

The following is the ESD cell file format:

```
CELL_1
CELL_2
...
CELL_n
```

Data_type: string, optional

`-esd_device_list {device1 device2 ....deviceN}`

Specifies the list of ESD devices. The ESD device information is used for accurate ESD analysis as it takes into account the internal grid from the ports to the ESD devices. The followings inputs are required for ESD tagging during macro PGV generation:

- an xDSPF file
- ESD device list

Data_type: string, optional

`-esd_parameters_file filename`

Specifies resistance information for the following types of ESD cells during ESD cell characterization:

- Diodes
- RC Clamp

When this parameter is specified, the software uses these electrical parameters from the generated PGV for ESD analysis.

The format of the input file is:

```
Cell <cell name>  <Ron> <Roff> <power_pin> <ground_pin> <voltage>
Type <esd cell type> <Ron> <Roff> <power_pin> <ground_pin> <voltage>
```

Here,

- `Cell` and `Type` are keywords
- *<cell name>* specifies the ESD cell name
- *<esd cell type>* specifies the ESD cell type, such as DIODE and CLAMP
- `RON` is the resistance of the complete cell when it is switched on.
- `ROFF` is the resistance of the complete cell when it is switched off.
- *power_pin* is the name of the power pin connected to the power net
- *ground_pin* is the name of the ground pin connected to the ground net
- *voltage* is the threshold value between the power and ground pin.
- When you specify the cell name, the RON and ROFF would be applied only for that cell, and when you specify the cell type, the RON and ROFF would be applied for all the ESD cells of the particular type.

The following is a snippet of the input file:

```
Cell HEADER_SWITCH 10 5 pwr_pin gnd_pin 0.4
```

Data_type: string, optional

`-esd_pin_list {`*pin1 pin2 ....pinN*`}`

Specifies the list of ESD pins. This parameter tags the user-specified pin as the ESD pin before running the `report_esd_network` command, and ensures that the software traces only this ESD power pin to bump and extracts resistance. The ESD cell pins that are tagged using this parameter are listed in the `.report` file.

Data_type: string, optional

`-exclude_tap_region_file` *filename*

Specifies to exclude certain tap points during macro PGV generation. This parameter helps to avoid false IR-drop violations inside macros caused by inadvertently assigning evenly-distributed tap current to certain undesired tie high/low paths or shielding paths at certain via layers. The Spice netlist and models are not mandatory parameters for excluding the tap points in the macro PGV generation flow.

The format of the region file is:

```
REGION X1 Y1 X2 Y2
REGION X3 Y3 X4 Y4
```

where, `REGION` keyword specifies that the coordinates following it is a tap current region. The unit for coordinates is micron.

Data_type: string, optional

`-extraction_command_file` *file*

Specifies the file name of an extraction command file.

Data_type: string, optional

`-followpins_tap_layer {lowest_lef_pin_layer | highest_lef_pin_layer | all_lef_pin_layers}`

Specifies the LEF metal layer used to generate current taps. This parameter has two arguments:

- `lowest_lef_pin_layer`: The tool generates current taps only on the lowest LEF metal layer. For example, if both METAL1 and METAL2 metal layers exist for the power/ground pin in the cell LEF, the tool uses METAL1 layer as the tap layer.
- `highest_lef_pin_layer`: The tool generates current taps only on the highest LEF metal layer. For example, if both METAL1 and METAL2 metal layers exist for the power/ground pin in the cell LEF, the tool uses METAL2 layer as the tap layer.
- `all_lef_pin_layers`: The tool generates current taps on all LEF metal layers. For example, if both METAL1 and METAL2 metal layers exist for the power/ground pin in the cell LEF, the tool uses both METAL1 and METAL2 layers as the tap layer. This is the default value.

Data_type: enum, optional

`-followpins_interface_node_layer {lowest_lef_pin_layer | highest_lef_pin_layer | all_lef_pin_layers}`

Specifies the LEF metal layer used to generate interface nodes. This parameter has three arguments:

- `lowest_lef_pin_layer`: The tool generates interface nodes only on the lowest LEF metal layer. For example, if both METAL1 and METAL2 metal layers exist for the power/ground pin in the cell LEF, the tool will only generate the interface nodes on the METAL1 layer.
- `highest_lef_pin_layer`: The tool generates interface nodes only on the highest LEF metal layer. For example, if both METAL1 and METAL2 metal layers exist for the power/ground pin in the cell LEF, the tool will only generate the interface nodes on the METAL2 layer.
- `all_lef_pin_layers`: The tool generates interface nodes on all LEF metal layers. For example, if both METAL1 and METAL2 metal layers exist for the power/ground pin in the cell LEF, the tool generates the interface nodes on both METAL1 and METAL2 layers. This is the default value.

Data_type: enum, optional

`-generate_bulk_pin_cap_separately {true | false}`

Specifies to generate separate capacitance values for body bias pins in the library report. This parameter allows you to check the bulk pin capacitance based on the Spice simulation. The default value of this parameter is `false`. When set to `false`, the software would sum up the capacitance values for the body bias pins and power/ground pins.

The following is a snippet of the library report:

```
cell_A
            VPP(0.9350)     2.3e-16         EARLY(1 taps) EM(1 taps) IR(1 taps)
            VBB(0.0000)     2.3e-16         EARLY(1 taps) EM(1 taps) IR(1 taps)
            VDD(0.9350)     8.1e-15         EARLY(1 taps) EM(1 taps) IR(1 taps)
            VSS(0.0000)     8.1e-15         EARLY(1 taps) EM(1 taps) IR(1 taps)
Cell Type = STDCELL                    Cell_Status: PASS
```

Here, the blue-highlighted text shows the capacitance values for the `VPP` and `VBB` body bias pins.

Data_type: enum, optional

`-generate_gray_box_data {true | false}`

Controls the generation of gray box data and storage of this data in the cell library. It is recommended to set this parameter to `false` to reduce the size of the PGV significantly (from 10MB to 1MB) without loss of accuracy. The default value of this parameter is `true`.

This parameter is applicable only for macro PGV generation (`-cell_type macros`).

`-generate_interface_node_x_direction {true | false}`

When the LEF port geometry is bigger than the tap node distance (`set_advanced_pg_library_mode -tap_node_distance`), multiple interface nodes are generated. You can specify this parameter to generate the interface nodes along the X direction.

The default value is `true`. When set to `false`, interface nodes are not generated along the specified direction.

Data_type: enum, optional

`-generate_interface_node_y_direction {true | false}`

When the LEF port geometry is bigger than the tap node distance (`set_advanced_pg_library_mode -tap_node_distance`), multiple interface nodes are generated. You can specify this parameter to generate the interface nodes along the Y direction.

The default value is `true`. When set to `false`, interface nodes are not generated along the specified direction.

Data_type: enum, optional

`-generate_interface_nodes_at_via_layer {true | false}`

Specifies to generate interface nodes at via locations in scenarios where the tap current source is a VIA layer.

Data_type: enum, optional

`-ignore_pg_nets {{cellname netname}+}`

Skips the specified power or ground nets in PGV generation.

Data_type: string, optional

`-import_xdspf_list_file filename`

Specifies the name of the xDSPF (third-party netlist) parasitic file that is imported by the Library Characterization engine to create PGDB. You must also specify the mapping file using the `-xdspf_layer_map` parameter.

When you specify this parameter, the following parameters are not supported:

- `-xtc_command_file`
- `set_pg_library_mode -spice_subckts`
- `set_pg_library_mode -gds_files`
- `set_pg_library_mode -gds_layer_map`
- `-techgen_dir`

Data_type: string, optional

`-input_port_value_list {pin1 value1 pin2 value2 ....pinN valueN}`

Specifies the voltage conditions of the signal inputs during power-grid library characterization. A typical usage model of this parameter include setting the SET or RESET signal port to be the same voltage as supply or ground (tie high/low) to obtain the correct model of operation.

The following example illustrates the use model:

```
set_advanced_pg_library_mode -input_port_value_list { RESETVDD 3.3 }
```

Data_type: string, optional

`-interface_node_location_file filename`

Specifies a file containing the location of interface nodes where tap currents are to be generated. The specified file includes the cell name, net name, layer name, and location of the nodes. This parameter allows to generate current taps at user-specified locations during PGV generation.

The format of the file is:

```
CELL <cell_name>
NET <net_name>
LAYER <layer_name> (x1,y1),(x2,y2)
```

Example:

```
CELL cellA
NET VDD
LAYER M0 (0,0.15),(0.585,0.177),(0.2,0.15),(0.3,0.177)

NET VSS
LAYER M0 (0.485,0.019)

CELL cellB
NET VDD
LAYER M0 (0.49,0.008),(0.24,0.008)
```

This parameter is applicable only for technology and standard cell PGV generation (`-cell_type techonly | stdcells`).

`-lef_layer_ignore_list {lef_layername+}`

Specifies the names of the LEF layers that power-grid library generation is to ignore when it creates the cell library.

Data_type: string, optional

`-lef_layer_ignore_list_file filename`

Specifies the name of a file containing a list of the LEF layers that power-grid library generation is to ignore when it creates the cell library. Each line in the file must consist of a single LEF layer name.

The format of the file is:

```
layer1
layer2
...
layern
```

Data_type: string, optional

`-lef_pin_short {true | false}`

Enables shorting of the unconnected LEF pins and ports to establish electrical connectivity. When set to `true`, shorts the interface nodes for all the cells that are being processed.

Default: `false`

Data_type: enum, optional

`-lef_pin_short_cell_file filename`

Specifies the name of a file containing a list of cells and the net name, metal layer name, and resistance value for each layer. It is used to short the LEF pins and ports by the user-specified resistance value. When this parameter is specified, only the given list of nets are shorted with the user-defined resistance values. If a resistance value is not specified for a layer, the default resistance value of 0.001 is used.

The format of the file is:

```
CELL <cell_name>
<net_name> <layer_name> <resistance_value>
```

Example:

```
CELL bump_70N_M4
VDD Metal4 12
CELL bump_70N_M5
VDD Metal4 15
VDD Metal5 12
```

`-lef_pin_short_cell_list {cell1 cell2 ...}`

Specifies a list of cells for which the the LEF pins and ports are shorted by the default resistance value of 0.001ohm. When this parameter is specified, all the nets of the given cells are shorted using the resistance value of 0.001ohm.

`-pg_library_generation_command_file file`

Specifies the file name of a power-grid library generation command file.

Data_type: string, optional

`-macro_parasitic_file filename`

Specifies the name of the SPEF file that provides signal net parasitics while creating a detailed dynamic view in Library Characterization. The SPEF should be extracted in the decouple_rc mode.

Note: A SPEF file is required only when you specify your own netlist in the Netlist-based flow. The SPEF file is a requirement when the netlist does not have signal caps. You should not specify the signal parasitics using the -ultasim_include_file option.

Data_type: string, optional

`-marker_layer_map filename`

Specifies the GDS layer mapping file containing the list of marker GDS layers that are used for layer operations during XTC processing. Marker layers are layers which do not have corresponding tech layers.

This parameter is used to support extraction of Metal-Insulator-Metal Capacitor (MIMCAP) geometries during power-grid library characterization. The following is the format of the mapping file:

```
Layer Type   marker layername   GDS keyword    gds layer number
metal          CTM_O             gds            77 1
metal          CBM_O             gds            88 1
```

Data_type: string, optional

`-pgdb_list_file filename`

Specifies the name of the PGDB header file containing the list of cells for which Library Characterization should generate power-grid views. The syntax of the file is:

CELL *cell_name*

PGDB *pgdb_directory*

The PGDB directory contains the PGDB for all the power and ground nets of the cell. The PGDB directory has cellname_Netname.* files.

When you specify this parameter, the following parameters are not supported:

- `-xtc_command_file`
- `set_pg_library_mode -spice_subckts`
- `set_pg_library_mode -gds_files`
- `set_pg_library_mode -gds_layer_map`
- `-techgen_dir`

Data_type: string, optional

`-pgdb_layer_map` *filename*

Specifies the name of the file containing the mapping between the layer name used in the technology file and the layer name used in the PGDB.

Data_type: string, optional

`-power_switch_characterization_voltages { val1 val2 val3 .... }`

Supports characterization of power gates at multiple voltages. You can specify multiple values of voltages so that the library characterization engine will characterize the power gates at each of the voltages. The software will then be able to get the characterized cell from the cell library for a particular voltage.

Data_type: string, optional

`-power_switch_multi_enable_characterization_method {enable_all | enable_single}`

Specifies the Ron/Idsat characterization method for power gate cells with multi-enable pins. This parameter allows you to control the generation of RON and Id-Vd curve for each enable pin. The possible characterization methods are:

- `enable_all`: All the enable pins are switched ON for Ron/Idsat characterization. The generated PGV model will contain a single Ron resistor with the total equivalent resistance value. The cell library report displays the equivalent Ron value of all the enable pins.
- `enable_single`: Only one enable pin is switched ON at a time for Ron/Idsat characterization, and the other pins are switched OFF. The generated PGV model will contain total number of Ron resistors that is equal to the number of enable pins. The cell library report displays Ron values separately for each enable pin.

This parameter is supported only for standard cells when the power gate parameters are specified using the `-power_switch_parameters_file` parameter. For multi-enable power gate cells, Ileak/Roff are always calculated by keeping all the enable pins OFF.

Default: `enable_all`

Example:

```
set_pg_library_mode ... -power_switch_parameters_file ./pg1.txt
set_advanced_pg_library_mode -power_switch_multi_enable_characterization_method enable_all
```

Data_type: string, optional

`-process_bulk_diffusion_ports {true|false}`

Specifies to create the `DIFF` layer ports of body bias pins in PGV.

By default, library generation ignores ports on the diffusion layer. In order to process body bias pin's ports on the diffusion layer for electrical analysis on body bias network, set this parameter to `true` during library generation.

Data_type: enum, optional

`-quit_on_extraction_errors {true|false}`

Specifies whether the software should exit with an error message or continue when it encounters extraction errors. The default is `true`.

Data_type: enum, optional

`-remove_em_view_dangling_resistor {true|false}`

Specifies to restore or remove dangling resistors from power-grid views. It is recommended to minimize dangling resistors for optimal analysis of IR drops and accurate Blech length calculation at the top level in EM analysis.

The default value is `false`.

Data_type: enum, optional

`-retain_generated_pgv_on_error {true | false}`

Specifies to retain an incomplete cell library created during PGV generation even if the tool exits with a fatal error. When set to `false`, the tool deletes the incomplete cell library. This parameter is recommended for use during initial library creation and debugging.

The default value is `false`.

Data_type: enum, optional

`-redo_characterization_count` *`count`*

Specifies to rerun the cells that failed due to machine issues during PGV library generation. This parameter allows you to specify the number of times the failed cells can be resubmitted in the distributed processing mode.

Default: `0`

Data_type: int, optional

`-on_resistance_measure_threshold` *`value`*

Specifies the switched pin voltage during Ron simulation. The specified value will be multiplied by the VDD voltage (<value>*vdd_vol) and applied to the switched pin during Ron simulation. The default value is `0.99`. You can specify a value between 0 and 1.

Data_type: float, optional

`-schematic {true|false}`

Specifies whether the subcircuit netlist is a schematic netlist.

Data_type: enum, optional

`-skip_switch_net_extraction {true | false}`

Specifies to skip extraction of switched nets inside the GDS file for a fine-grain memory cell. The software will add the capacitance from the switched net to the always-on net. The software will thus consider the cell as a regular memory without switched/power-gated nets. When this parameter is specified, you do not need to specify the port file (`set_advanced_pg_library_mode -add_port_labels`) as the switched nets are not traced. The final PGV will be created with the always-on and VSS nets.

The power-gate library report file will contain information about the missing switched net, and the capacitance and current that was contributed by the switched net.

Data_type: enum, optional

`-source_location_file {`*`filename`*`}`

Specifies the name of the file containing the locations of the voltage sources to be connected to the pad cells. For more information on the source location file, see Source Location File Format.

Data_type: string, optional

`-spectre_path` *`value`*

Specifies an external Spectre path for PGV generation. This parameter allows you to use a different Spectre version than the one embedded with the software.

Data_type: string, optional

`-stdcell_characterization_voltage_value {`*`val1 val2 val3 ...`*`}`

Specifies capacitance characterization voltages when generating standard cell PGVs containing voltage-dependent capacitance table for the specified cell. This parameter allows you to compute capacitance based on the user-defined voltage values. The `-stdcell_characterization_voltage_value` parameter must be specified with the `set_pg_library_mode -enable_multi_voltage_cap_generation true` parameter.

If this parameter is not specified with `-enable_multi_voltage_cap_generation`, capacitance is calculated based on the software-derived voltage range.

When the `-stdcell_characterization_voltage_value` parameter is specified, the library generation engine prints the capacitance values and corresponding voltages in the report file, and the rail analysis engine prints the capacitance used for each instance.

The following example illustrates the use model of this parameter:

```
set_pg_library_mode  -ground_pins VSS -power_pins  {VDD 0.9 TVDD 0.9 } -cell_type stdcells \
...
-enable_multi_voltage_cap_generation true
set_advanced_pg_library_mode -stdcell_characterization_voltage_value {1.0 1.2 1.5 2.0 3.3}
```

Data_type: string, optional

`-strict_input_check {true|false}`

Specifies that the software should not create any view for the cell if the EM view is not generated. When set to `true`, the software displays an error message if the Spice netlist or GDS is not specified, and gives details about the cell name and the reason for not generating the view. The default value of this parameter is `false`.

Data_type: enum, optional

`-tap_node_distance` *value*

Specifies to set the maximum distance between interface nodes when they get generated in power-grid library generation. The default is `50` microns.

Library generation creates interface nodes on port geometries in LEF and Extraction, uses them for top level connectivity tracing and also to connect the cell/block onto top level routing. The interface node generation is purely based on port geometries (tap points are placed one per port). For ports greater than `-tap_node_distance`, they are placed at `-tap_node_distance` apart. If the port pin geometry in LEF is bigger than the setting of the `-tap_node_distance` parameter, at least two nodes will get generated per port geometry.

Data_type: float, optional

`-techgen_dir` *directory*

Specifies the techgen directory.

Data_type: string, optional

`-thunder_command_file` *file*

Specifies the file name of a thunder command file.

Data_type: string, optional

`-trim_metal_layer_map` *file*

Specifies the name of the file containing the mapping between a metal layer and its corresponding trim layer. When this parameter is specified, the trim layer shapes are automatically removed from the routing layer shapes to correctly extract all the nets in the generated PGV.

File Format:
metal_layer_name trim_layer_name

```
M0 CM0
M1 CM1
M2 CM2
```

Data_type: string, optional

`-use_embedded_spectre {true | false}`

When set to `false`, allows you to specify a different Spectre version than the one embedded with the software. Library Generation will automatically pick Spectre from users' path.

Default : `true`

Data_type: enum, optional

`-use_peak_current_distribution {true | false}`

Default: `false`

Specifies to use the maximum currents of devices in the Spice netlist for current distribution in macros. When set to `true`, enables peak current distribution for faster PGV generation. The default (`false`) behavior is to enable activity propagation based current distribution.

Data_type: enum, optional

`-verbosity {true | false}`

Controls the amount of information displayed in the console and written to the log files during power-grid library generation.

Default: false

You can use this parameter for detailed troubleshooting and controlling the verbosity level in the log file.

Data_type: enum, optional

`-via_based_current_config_file filename`

Specifies the name of the configuration file containing tap current details for location-based via current generation. If `-enable_via_based_current_distribution` is set to `true`, this parameter must be specified.

Configuration File Format:

```
#------------------------------------------------------------------------
CURRENT_UNIT <default_value>
INPUT_SLEW <default_value>
OUTPUT_SLEW <default_value>
MAX_DISTANCE <Threshold distance for snapping via tap if the exact xy location is not found as specified>
DELAY <default_value>
PERIOD <default_value>
INTRINSIC_CAP { power_net cap ground_net cap}
#------------------------------------------------------------------------
ID     PGPIN_NAME     X    Y     AVG_CURRENT     PEAK_CURRENT
#------------------------------------------------------------------------
```

Configuration File Example

```
#------------------------------------------------------------------------
CURRENT_UNIT mA
INPUT_SLEW 20ps
OUTPUT_SLEW 10ps
MAX_DISTANCE 2.5
DELAY 5ps
INTRINSIC_CAP { VDD 40nF VSS 5nF }
#------------------------------------------------------------------------
ID      PGPIN_NAME     X          Y     AVG_CURRENT     PEAK_CURRENT
#------------------------------------------------------------------------
PVDD_1   VDD       16.3090   7.6470     50.0373       100.037
PVDD_2   VDD       16.6900   7.6470     50.0373       100.037
PVDD_3   VDD       17.0700   7.6470     50.0373       100.037
...
```

When this feature is enabled, the tool by default generates a report "`dropped_via_location_taps.txt`" in the output directory to give a list of via IDs that were dropped because they were not within the snapping threshold limit. By default, the snapping threshold is zero. It is controlled by "`MAX DISTANCE`" in the configuration file.

Data_type: string, optional

`-well_capacitance_file filename`

Specifies the name of a file containing a list of cells and the well capacitance value for each.

Data_type: string, optional

`-xdspf_layer_map filename`

Specifies the name of the xDSPF mapping file containing layer type and layer stack information from bottom most layer to top most.

You must define layers in the stack from top most layers at the top of the stack to bottom most layers at the bottom of the stack. With reference to this layermap stack, library generation will create the PGDB for the layers as defined in the xDSPF layermap file.

Data_type: string, optional

`-xtc_command_file filename`

	Specifies the name of the XTC command file. *Data_type:* string, optional
`-xtc_include_file` *filename*	
	Specifies the file name of an XTC include file used during characterization. For example, you can include XTC commands to ignore GDS cells, to blackbox lower-level hierarchy cells, or to perform some basic GDS layer operations. *Data_type:* string, optional

Examples

- The following command creates a power-grid library:
  ```
  set_advanced_pg_library_mode \
  -pg_library_generation_command_file library_inc.cmd
  -thunder_command_file simulator.cmd
  ```
- In the following example, power-grid library generation ignores the LEF *metal1* and *metal3* layers when it creates the cell library:
  ```
  set_advanced_pg_library_mode -lef_layer_ignore_list {metal1 metal3}
  ```
- The following line indicates that a file named *nolef.txt* contains a list of the LEF layer names that power-grid library generation should ignore:
  ```
  set_advanced_pg_library_mode -lef_layer_ignore_list_file {nolef.txt}
  ```
- The following commands tags the `VDD` and `TVDD` pins as the ESD pins:
  ```
  set_advanced_pg_library_mode -esd_cells {IOESD} -esd_pin_list {VDD TVDD}
  ```
- The following command specifies the ESD device `dev1` and the xDSPF file `cell1.xdspf` to be used for ESD tagging:
  ```
  set_advanced_pg_library_mode -import_xdspf_list_file cell1.xdspf -esd_cells {IOESD} -esd_device_list {dev1}
  ```
- The following example specifies that FOREIGN statements are to be assumed for cells with no FOREIGN statements and that the FOREIGN cells should be placed at the inverse of the origin:
  ```
  -assume_foreign true
  -assume_foreign_mode 1
  ```
- The following example specifies that FOREIGN statements are to be assumed for cells with no FOREIGN statements, and the FOREIGN cells should be placed at (0,0):
  ```
  -assume_foreign true
  -assume_foreign_mode 0
  ```

Source Location File Format

The voltage source location file format is as follows:

```
<MIN_SPACING spacing_value>
<DENSITY density_value>
CELL cell_name
<[PORT|DETAILED] {>
NET net_name
<[PORT|DETAILED] {>
lef_layer_name <PORT|DETAILED> x y [x y]
lef_layer_name <PORT|DETAILED> x y [x y]
<}>
<}>
This source location file syntax includes the following parameters:
```

- NET net_name
  These specifications apply to the voltage source locations that follow until Library Generation encounters a new CELL or NET line. You can specify voltage sources as single points or as rectangles. If you specify them as rectangles, Library Generation generates a grid of voltage sources inside these rectangles.
- PORT | DETAILED
  The source location file can specify voltage sources to be used with port or detailed views.

lef_layer_name <PORT|DETAILED> x y [x y]
Lines beginning with a LEF layer name specify the actual location of the voltage sources.

You can specify this location as a single point by providing one set of values for x and y. These x and y coordinates are specified in microns and use the same coordinate system as that used for LEF ports in the LEF file.

Alternatively, you can specify the voltage source locations as a rectangle by giving two sets of x and y locations. In this case, Library Generation attempts to generate a 10-by-10 grid of voltage sources inside this area. From a rectangle specification, it does not create voltage sources closer than 1 micron apart, so if the rectangle is less than about 10 microns on a side, it generates fewer voltage sources.

In the source location file, the location of the voltage sources is specified by using a LEF layer name and location in LEF coordinate space. However, because of the way LibGen uses this data internally, no LEF geometries are actually required to exist at the location where the voltage sources are to be placed. Library Generation uses this data internally with the layout of the block (that is, the GDS data) during detailed power-grid view generation, so it only requires that there be layout (GDS) geometries at the location of the voltage sources. Similarly, the LEF layer name used in the source location file does not actually have to be included in any of the input LEF files; it must be defined only in the layer map used by Library Generation.

This is an example of the contents of the voltage source location file:

```
CELL BUMP
NET vcc
METAL1 11.0 16.0
METAL1 0.0 15.0 22.1 17.5
```

Here is an example file using MIN_SPACING and DENSITY to create a 10 x 10 grid of voltage sources on the VSS net in the detailed view of the NAND1 cell:

```
DENSITY 10
MIN_SPACING 1
CELL BUMP
DETAILED
NET VSS
```

Related Information

- set_pg_library_mode
- write_pg_library
- rename_pg_library
- "Power-Grid Library Generation" in the *Voltus Stylus Common UI User Guide*

set_pg_library_mode

```
set_pg_library_mode
[-help]
[-bulk_power_pins { pin1 voltage1 ... pinN voltageN }]
[-bulk_ground_pins { ground_pin_list }]
[-decap_cells_file filename ]
[-cells_file filename ]
-cell_type { techonly | stdcells | macros }
[-current_distribution {propagation | dynamic_simulation trigger_file | current_region region_file}]
[-decap_cells cell_name_list ]
[-default_area_capacitance value ]
[-default_power_voltage value ]
-extraction_tech_file file
[-filler_cells cell_name_list ]
[-stream_files file_list ]
[-stream_layer_map file ]
[-generate_port_using_stream mbb_layer_name ]
[-ground_pins { ground_pin_list }]
[-non_zero_ground_pins { pin1 voltage1 ... pinN voltageN } ]
[-power_pins { pin1 voltage1 ... pinN voltageN } ]
[-layer_tap_node_distance { {LayerName1 tap_node_distance} {LayerName2 tap_node_distance} ... }]
[-layers_to_be_ignored_during_tap_creation {layername+}]
[-layers_to_have_ports_created {layername+}]
[-lef_layer_map file ]
[-power_switch_fine_grain_simulation {true|false} ]
[-power_switch_parameters  {{ cell supply switched Ron Idsat Ileak Control_Pin_Logic} + } ]
[-spice_corners corner_list]
[-spice_models file_list]
[-spice_subckts file_list ]
[-spice_subckts_xy_scaling value]
[-stop_via layername ]
[-temperature temp_value_in_degree_celcius ]
[-liberty_files file_list]
[-custom_label label]
[-macros_config_file filename]
[-enable_distributed_processing  {true|false} ]
[-spice_netlist_file filename]
[-exclude_cells_file filename]
[-use_power_switch_data_from_lib {true|false}]
[-enable_multi_voltage_cap_generation {true | false}]
[-fine_grain_memory_cell  {true | false} ]
[-internal_power_pins power_pin_list]
[-internal_ground_pins ground_pin_list]
[ -power_switch_parameters_file filename ]
[-ignore_stream_cells {gds_cell_name_list}]
[-enable_device_size_current_distribution {true | false}]
[-calculate_metal_layer_density {true | false}]
[-max_metal_density_tile_dimension {maxlength maxwidth}]
[-add_power_pin_location_cells cell1 cell2 ...]
[-cell_power_pin_location_details filename]
[-tap_node_distance_file filename]
[-design_qrc_layer_map file ]
[-tech_stream_layer_map filename]
[-pin_shape_current_distribution filename]
```

Specifies what kind of power-grid library will be created. This command is used to define the type of input and configuration for generating power-grid libraries. You must specify this command before running the `write_pg_library` command.

Parameters

`-help`	
	Outputs the command usage.

`-add_power_pin_location_cells cell1 cell2 ...`

Specifies the name of the low-dropout (LDO) cell for which a voltage source is to be added. You can specify a list of cells. This parameter is applicable to the LDO cells only, and must be used when creating voltage sources for these cells. When this parameter is specified, all the output pins will be made voltage sources using the highest voltage value of the cell.

The `-add_power_pin_location_cells` parameter is used with the `-cell_power_pin_location_details` parameter.

`bulk_ground_pins { ground_pin_list }`

Specifies the names of nets that connect transistor bulk terminals to the ground net. This information is used to prevent any current from being assigned to these nets. If a net is connected to any transistor terminals other than a bulk terminal, you must specify that net with the `-ground_pins` parameter.

Data_type: string, optional

`-bulk_power_pins { pin1 voltage1 ...pinN voltageN }`

Specifies the names of nets that connect transistor bulk terminals to the power net. This information is used to prevent any current from being assigned to these nets. If a net is connected to any transistor terminals other than a bulk terminal, you must specify that net with the `-power_pins` parameter.

Data_type: string, optional

`-calculate_metal_layer_density {true | false}`

Enables metal layer density calculation for macros. The default value is `false`.

When this parameter is set to `true`, Library Generation calculates and stores the area density information for each metal layer in macros for accurate thermal analysis. Metal density for macros is calculated by dividing the macro cell into tiles, and computing the area of polygons in each tile. The calculated metal density values are used in the Power Analysis flow to generate the metal power density map. This power map file is later used as input by Celsius Thermal Solver, the thermal analysis tool for packages and boards.

This parameter is applicable only for macro PGV generation (`-cell_type macros`).

Data_type: enum, optional

`-cells_file filename`

Specifies a file containing a list of cells that will be processed.

The `-cells_file` parameter is required when `set_pg_library_mode -cell_type macros` is selected. The program spawns the library generation job in serial mode for each macro in the cell list file. A separate library for each macro is generated. This facilitates debugging when library generation of any macro fails. It also helps in improving the overall performance. The following is a sample cell list file:

```
CELL1
CELL2
CELL3
```

Data_type: string, optional

`-cell_power_pin_location_details filename`

Specifies a file that contains all the information required to add a voltage source at the desired location for the power pad and LDO cells. This file includes the pad cell name, pin name, layer name, and domain name.

The format of the file is:

```
cell_name      pin_name    layer_name  domain_name
PADCELL           VDD       Metal3       VDD
```

- *pin_name* is the power/ground name for pad cells, and is the output pin name for LDO cells. The pin name could be represented by using the wildcard character "*" to indicate that voltage sources are to be added to all pins of a power pad/LDO cell.
- *domain_name* is the pin name for pad cells. For LDO cells, it is the power or ground name whose voltage needs to be used.

Example:

```
set_pg_library_mode \
    -ground_pins                {VSS 0 ExtVSS 0} \
    -power_pins                 {VDD 1.2 ExtVDD 1.2} \
    -cell_type                   techonly \
    ...
    -add_power_pin_location_cells             {CELL1 CELL2 CELL3} \
    -cell_power_pin_location_details          /VoltageSource/config.txt \
```

The following is a snippet of the config.txt file:

```
CELL1 VDD Metal3 VDD
CELL1 VSS Metal2 VSS
CELL1 Z Metal2 VDD
CELL2 Z Metal3 VSS
CELL3 * VDD
```

`-cell_type { techonly | stdcells | macros}`

Specifies the type of library creation.

- `techonly` – Creates a technology Library that contains the tech file, area capacitance specification, decap/filler/powergate cells, and Tech view for all the cells. A technology library is a requirement for running rail analysis.
- `stdcells` – Generates a power-grid library for all the standard cells using the Spice netlist. The stdcells power-grid library does not contain the tech file. Power-grid is connected at the LEF pin, and all the capacitance are derived from simulation. The power-grid library contains three kind of power-grid views (PGVs), Early/IR/EM, and the views are selected based on the accuracy mode of rail analysis.
- `macros` – Generates a power-grid library for memory, I/Os and macros using the Spice and GDS netlist. GDS is mandatory for the cell type macros. The `macros` power-grid library does not contain the tech file. Power-grid is connected down to contact/specified via, and all the capacitance are derived from simulation. The power-grid library contains three kind of PGVs, Early/IR/EM, and the views are selected based on the accuracy mode of rail analysis.

Data_type: enum, required

`-current_distribution {propagation | dynamic_simulation` *`trigger_file`* `| current_region` *`region_file`*`}`

Specifies the method of current distribution inside the cell.

- **Propagation** uses Voltus library simulator to determine the propagation activity to distribute the current.
- **Dynamic simulation** uses a trigger file to run detailed analysis to determine current distribution. For more information on the format, see Trigger File Format.
- **Current region** specifies the amount of current to distribute within a cell region. For more information on the format, see Current Region File.

Data_type: string, optional

`-custom_label` *`label`*

Saves the specified label in the PGV for easy identification. This label can be used for revision control, label for Spice netlist/model, and internal keywords. It will be stored as “Custom Label” in VERSION.TXT in the .cl.

Data_type: string, optional

`-decap_cells` *`cell_name_list`*

Specifies the names of cells that have explicit decoupling capacitance for use in decap optimization. These cells are nearly equivalent to feedthrough cells, except that there is typically an active device between the power and ground nets to provide extra decoupling capacitance. Power-grid view library generation uses this list of cells to ensure that a capacitance value is associated with each cell. If no capacitance is associated with a cell, a warning message will be issued.

This parameter supports wildcards (*).

Data_type: string, optional

`-decap_cells_file` *filename*

Specifies the name of a file containing a list of cells and the decoupling capacitance value for each. You can also specify the series resistance value along with the decoupling capacitance value to accurately model the cell decoupling capacitance. The cell decap file format allows you to specify capacitance for multiple voltage values to support multi-voltage capacitance simulation for standard cell and macro PGV generation.

File Format:

```
UNIT <X>
CELL cell_name
PIN pin_name capacitance resistance

or
UNIT <X>
CELL cell_name
VOLTAGE value1
PIN pin_name capacitance resistance
...
....

VOLTAGE value2
PIN pin_name capacitance resistance
...
...
```

Unit is optional, default is `1e-15`. The *value* for each cell will be multiplied by the unit, except if engineering notation is provided.

Examples:

```
UNIT 1
CELL cell1

VOLTAGE 1.0
PIN VDD  1e-18
PIN VSS  1e-18

VOLTAGE 0.9
PIN VDD  1e-18
PIN VSS  1e-18
```

Data_type: string, optional

`-default_area_capacitance` *value*

Specifies the default amount of area based decoupling capacitance in a cell. The default area-cap is 0.0 (unit is fF per micron-square). You must provide a value for area-based decoupling capacitance.

Data_type: string, optional

`-default_power_voltage` *value*

Specifies voltage for the power pins which are present in LEF but not specified using the `-power_pins` parameter. As a result, the `-power_pins` parameter of the command is optional. The default voltage value specified using `-default_power_voltage` is added to all power pins of the cell.

If both the parameters `-power_pins` and `-default_power_voltage` are not specified, the default voltage of 1.2 will be used.

Data_type: string, optional

`-design_qrc_layer_map` *file*

Specifies a file that provides the mapping of the layer names in the LEF design file to the layer names in the Quantus technology file.
File Format:

```
lefdef_layername    technology_layername
M1                   METAL_1
VIA1                 VIA_1

M2                   METAL_2
VIA2                 VIA_2
...
```

`-enable_device_size_current_distribution {true | false}`

Specifies to compute the tap current distribution for a macro based on device sizes (Width, Length), without performing any simulation. This parameter is used for macro PGV generation in a scenario when the Spice netlist is available but the Spice models are not available.

Default: `false`

Data_type: enum, optional

`-enable_distributed_processing {true|false}`

Specifies to enable parallel processing for the Library Generation flow.

Data_type: enum, optional

`-enable_multi_voltage_cap_generation {true | false}`

Specifies to compute capacitance values over a range of input voltages, including the nominal voltage. It generates 10 values from 0.1 to 1.3 times the vdd/vss voltage. The default value is `false`.

When set to `true`, it generates a PGV that contains the voltage-dependent capacitance table for the specified cell. This parameter allows you to use the accurate cell capacitance at different supply voltages in power-up analysis and in different voltage domains in dynamic analysis.

Data_type: enum, optional

`-exclude_cells_file` *`filename`*

Specifies a file containing a list of cells (one per line) which should be excluded from power-grid library generation.

Data_type: string, optional

`-extraction_tech_file` *`file`*

Specifies the name of the technology file that will be used for extraction.

Data_type: string, required

`-filler_cells` *`cell_name_list`*

List the names of the filler or feedthrough cells that have no GDS data. These cells can be swapped out for decaps during decap optimization.

This parameter supports wildcards (*).

Data_type: string, optional

`-fine_grain_memory_cell {true | false}`

Specifies to tag the cell as the fine-grain cell in the PGV library when characterizing the cell, and ensures that the software automatically uses this information to identify the fine-grain memory cells in the rail analysis flow. The default value is `false`.

This parameter is used with the `-internal_power_pins` and `-internal_ground_pins` parameters to specify the list of switch power/ground pins. It is recommended to use these parameters when characterizing fine-grain cells to avoid any potential cell type identification conflict in the rail analysis flow.

Data_type: enum, optional

`-generate_port_using_stream` *`mbb_layer_name`*

Identifies the layer name that library generation uses to compute the placement bounding box for GDSII cells that are placed by the DEF file.

Data_type: string, optional

`-ground_pins {` *`ground_pin_list`* `}`

Specifies the names of the nets considered ground nets for all cells for which the ground nets are not specified.

Data_type: string, optional

`-ignore_stream_cells {gds_cell_name_list}`

Specifies to ignore the specified cells during macro PGV generation. When this parameter is specified, the Library Generation engine does not extract data for these cells. This parameter supports wildcards.

Example:

The following command does not extract data for the `cell3` cell:

```
set_pg_library_mode -ignore_stream_cells {cell3}
```

Data_type: string, optional

`-internal_power_pins power_pin_list`

Specifies the list of switch power pins.

Data_type: string, optional

`-internal_ground_pins ground_pin_list`

Specifies the list of switch ground pins.

Data_type: string, optional

`-layer_tap_node_distance { {LayerName1 tap_node_distance} {LayerName2 tap_node_distance} ... }`

Default: None

Specifies tap node distances for certain layers. Using this parameter, you can customize the tap node distance for each layer during power-grid library generation.

The tap node distance is in micron. If the tap node distance is specified for a single layer, double brackets {{}} are still required.

Data_type: string, optional

`-layers_to_be_ignored_during_tap_creation {layername+}`

Specifies the names of the layers that library generation is to ignore when it creates the current taps. This parameter can be used to suppress the generation of any nodes or taps on layers that should not be extracted. Some technologies include layers that are only used to influence the extraction of other layers.

Data_type: string, optional

`-layers_to_have_ports_created {layername+}`

Specifies the names of the layers for which ports are to be created. This parameter can be used to create ports for the poly and diffusion layers in the technology file. By default ports are only created on metal layers.

Data_type: string, optional

`-lef_layer_map file`

Specifies a file that provides the mapping of layers between the LEF and technology file. This parameter is optional. Library Generation automatically generates the layermap file using the technology LEF (containing all layer and via definitions) and Quantus Tech (extraction library) files, therefore, a LEF layermap file is not a mandatory input. The automatically generated layermap file is located in the PGV temp directory.

The following is the format and sample content of the auto-generated file (`lefdefLayerMap_AutoGenerated.txt`):

```
Layer Type    tech_layername    lefdef    lefdef_layername
via               VIA_9         lefdef      VIA9
metal             METAL_1       lefdef      METAL1
```

You can specify this parameter if you want to provide your own layermap file.

Data_type: string, optional

`-liberty_files file_list`

Specifies to directly read decoupling capacitance and resistance from the specified Liberty file for standard cell and macro PGV generation.

The following is an example of the Liberty construct required for capacitance/resistance:

```
intrinsic_parasitic () {
  when : "I";

  intrinsic_capacitance (VDD) {
    value : 0.000248992;
  }

  intrinsic_resistance (VSS) {
    related_output : ZN;
    value : 1.89195;
  }

  intrinsic_capacitance (VSS) {
    value : 0.000639008;
  }
}
```

If there are multiple `when` conditions, the average intrinsic resistance/capacitance value is taken.

Data_type: string, optional

`-macros_config_file` *filename*

Specifies to provide configuration details for each macro cell per Library Generation run. This is an additional input for `macros` cell type wherein you can provide macro specific Spice and GDS. The format of the file is:

```
CELL <cell1>
GDS_FILE <gds_file1>
SUBCKT_FILE <subckt_file1>
DECAP_FILE <decap_file1>
TRIGGER_FILE <trigger_file1>
POWER_GATE_PARAMS supply <supplyName> switched <switchedName>
CELL <cell2>
GDS_FILE < gds_file1 >
SUBCKT_FILE < subckt_file1 >
```

`CELL` is the keyword to specify a new cell that is to be characterized. This is a mandatory keyword. This keyword is followed by optional keywords, like `GDS_FILE` and `SUBCKT_FILE`, that are specific to the cell.

Data_type: string, optional

`-max_metal_density_tile_dimension {`*maxlength maxwidth*`}`

Specifies the tile dimensions, wherein *maxlength* denotes maximum length and *maxwidth* denotes maximum width in µm. The default value for length or width is 50µm. The number of tiles in the X and Y direction is determined using the length and width dimension.

This parameter should only be used when the `-calculate_metal_layer_density` parameter is set to `true`.

Data_type: string, optional

`-non_zero_ground_pins { `*pin1 voltage1 ...pinN voltageN*` }`

Default: None

Specifies the name and voltage of the nets considered ground nets for all cells for which the elevated ground nets need to be specified. This parameter allows you to provide a list of non-zero ground be it positive or negative for the ground nets. If the same name has been specified for both the parameters `-non_zero_ground_pins` and `-ground_pins`, then `-non_zero_ground_pins` will be given priority.

This parameter is applicable only for standard cell and macro PGV generation (`-celltype stdcells | macros`).

Example:

```
set_pg_library_mode -non_zero_ground_pins {VSS -1.05 VSSA 0.08 VSSA1 1.09}
```

Data_type: string, optional

`-pin_shape_current_distribution` *filename*

Specifies a current distribution file that allows you to state how current should be distributed across multiple pin shapes of power and ground pins. Multiple pin shapes exist for VDD and VSS pins, and current drawn through each pin shape is non-uniform depending upon the output bit that is switching. This parameter allows you to define current distribution for multi-height multi-bit flip-flop (MBFF) characterization. The parameter allows you to accurately model the current through each pin shape.

File Format

```
CELL <cell name>
PGPIN VDD:(<LEF_layer_name>,<LEF_coordinates>):(<LEF_layer_name>,<LEF_coordinates>)
PGPIN VSS:(<LEF_layer_name>,<LEF_coordinates>):(<LEF_layer_name>,<LEF_coordinates>)
PIN <signal_pin_name> <current_ratio_for_VDD> <current_ratio_for_VSS>
```

Here,

- *<LEF_layer_name>,<LEF_coordinates>* specifies to assign the current ratio to the pin shape occupying the coordinates.
- *LEF_layer_name* the layer name to which the current taps are to be created

Example

```
CELL NAND3X8
PGPIN VDD:(Metal1,1,1):(Metal1,4,1)
PGPIN VSS:(Metal1,2,2):(Metal2,5,2):(Metal2,8,2)
PIN Q1 0.00:1.00 0.00:0.00:1.00
PIN Q2 0.90:0.10 0.95:0.05:0.00
```

Data_type: string, optional

`-power_pins { pin1 voltage1...pinN voltageN }`

Specifies the name and voltage of the nets considered power nets for all cells for which the power nets are not specified. If both the parameters `-power_pins` and `-default_power_voltage` are not specified, the default voltage of 1.2 will be used.

Data_type: string, optional

`-power_switch_fine_grain_simulation {true|false}`

Specifies to enable finegrain powergate simulation.

Data_type: enum, optional

`-power_switch_parameters{{ cell supply switched Ron Idsat Ileak Control_Pin_Logic} + }`

Specifies the information that technology library generation needs to characterize a powergate cell. It includes the cell name, unswitched power pin, switched power pin, on resistance in ohms, saturation current in milliamps, leakage current in milliamps, and control pin logic.

If ron, idsat, or ileak values are specified, they will be used to override the data extracted by Voltus library simulator.

`Control_Pin_Logic` can have the following values:

- `enableHigh` - control pin(s) of the control logic will be connected to the power net to turn on the power gate instances.
- `enableLow` - control pin(s) of the control logic will be connected to the ground net to turn on the power gate instances.

The control pin logic allows you to specify whether the enable pin of the specified power gate is active high or active low, thus ensuring that the appropriate voltages are applied to the enable pins.

This parameter also allows you to specify multiple functional modes to enable different combinations of control pins in each mode. For multiple functional modes, the tool calculates RON for each mode separately and generates a single PGV for the specified modes.

Format for Multiple Functional Modes:

```
set_pg_library_mode -power_switch_parameters { {cell_name mode_name1 {supply switch Ron Idsat Ileak Control_Pin_Logic} mode_name2
{supply switch Ron Idsat Ileak Control_Pin_Logic}... mode_nameN {supply switch Ron Idsat Ileak Control_Pin_Logic} }
```

Examples:

- For control pin logic:
  ```
  -power_switch_parameters {  {HEADER_SWITCH TVDD VDD enableHigh} }
  ```
- For Single Powergate:
  ```
  -power_switch_parameters{{HDRSID0 TVDD VDD}}
  ```
- For 2 Header Powergate:
  ```
  -power_switch_parameters{{HEAD16_A12TH_C35 VDDG VDD} {HEAD8_A12TH_C35 VDDG VDD}}
  ```
- For Multiple Footer Powergate
  ```
  -power_switch_parameters {{FOOTBUFVHSV16 VSSG VSS} {FOOTBUFVHSV32 VSSG VSS} {FOOTBUFVHSV64 VSSG VSS} {FOOTBUFVHSV8 VSSG VSS}}
  ```
- For Multiple Functional Modes:
  ```
  -power_switch_parameters { {HEADER_SWITCH S1 {TVDD VDD 750 .5 4.0e-8V} S3 {TVDD1 VDD 350 .3 5.0e-8V} S2 {TVDD VDD 450 .5 4.0e-8V}
  } {RING_SWITCH TVDD VDD 750 .5 4.0e-8} }
  ```

Data_type: string, optional

`-power_switch_parameters_file` *filename*

Specifies a file that contains details of power switches that have multiple control pins, input pins, and power nets. The control pins, input pins, and power nets can have either sweep voltages or user-defined voltages.

This parameter also allows you to specify multiple functional modes to enable different combinations of control pins in each mode. For multiple functional modes, the tool calculates RON for each mode separately and generates a single PGV for the specified modes.

File Format:

```
CELL <cell_name>
PGATE_PAIR  <AON_PIN> <SW_PIN> <CTRL_PIN> <Polarity|VoltageValues>
PGATE_PAIR  <AON_PIN> <SW_PIN> <CTRL_PIN> <Polarity|VoltageValues>
…
PWR_PIN <net_name> {voltage_values}
PWR_PIN <net_name> {voltage_values}
…
INPUT_PIN <pin_name> <Polarity|VoltageValues>
INPUT_PIN <pin_name> <Polarity|VoltageValues>
…
CHARACTERIZATION_VOLTAGES { val1 val2 val3 .... }
END
```

File Format for Multiple Functional Modes:

```
CELL <cell_name>
MODE_NAME mode_name1
PGATE_PAIR <AON_PIN> <SW_PIN> <CTRL_PIN> <Polarity|VoltageValues>
PWR_PIN <net_name> {voltage_values}
INPUT_PIN <pin_name> <Polarity|VoltageValues>
CHARACTERIZATION_VOLTAGES <VolageValues>
END_MODE
```

```
MODE_NAME mode_name2
PGATE_PAIR  <AON_PIN> <SW_PIN> <CTRL_PIN> <Polarity|VoltageValues>
PWR_PIN <net_name> {voltage_values}
INPUT_PIN <pin_name> <Polarity|VoltageValues>
CHARACTERIZATION_VOLTAGES <VolageValues>
END_MODE
...
END
```

- PGATE_PAIR and CHARACTERIZATION_VOLTAGES are mandatory keywords.
- PGATE_PAIR is the keyword to specify the always-on power pin, switched power pin, and control pin names, and the voltage values at which these pins are simulated. If you specify voltage values, then the software uses these user-defined values during power gate characterization. Alternatively, if you specify polarity (high/low), then the software uses the sweep voltage values (CHARACTERIZATION_VOLTAGES) during power gate characterization. If you do not specify both polarity or voltage values, then the software by default uses the sweep voltage values (CHARACTERIZATION_VOLTAGES) during power gate characterization.
- CHARACTERIZATION_VOLTAGES is the keyword to specify sweep voltage values. The number of voltage values specified with the PGATE_PAIR keyword must match with the number of voltage values specified with the CHARACTERIZATION_VOLTAGES keyword.
- PWR_PIN is the keyword to specify additional power pins of the cell other than the always-on and switched power pins.
- INPUT_PIN is the keyword to specify additional signal pins of the cell other than the control pins. If you specify voltage values, then the software uses these user-defined values during power gate characterization. Alternatively, if you specify polarity (high/low), then the software uses the sweep voltage values (CHARACTERIZATION_VOLTAGES) during power gate characterization. If you do not specify both polarity or voltage values, then the software by default uses the sweep voltage values (CHARACTERIZATION_VOLTAGES) during power gate characterization.

Examples:

Example 1: User-specified voltage values for control pins, input pins, and power nets

```
CELL cellA
PGATE_PAIR AONVDD SWVDD SLEEPN {0.600 0.660 0.700 0.750 0.800 }
PWR_PIN VDD_PS {0.7 0.8 0.9 1.0 1.1 }
INPUT_PIN PS_DISABLE {0.600 0.660 0.700 0.750 0.800 }
CHARACTERIZATION_VOLTAGES {0.600 0.660 0.700 0.750 0.800 }
END
```

Example 2: User-specified polarity

```
CELL cellb
PGATE_PAIR TVDD VDD SLEEPN
PWR_PIN VDD_SIDE
INPUT_PIN S1 HIGH
CHARACTERIZATION_VOLTAGES {0.600 0.660 0.700 0.750 0.800 }
END
```

Example 3: powergate_param_file.txt

```
CELL HEADER_SWITCH
MODE_NAME S1
PGATE_PAIR TVDD VDD NSLEEPIN LOW
CHARACTERIZATION_VOLTAGES {0.9 1.0 1.1}
END_MODE

MODE_NAME S2
PGATE_PAIR TVDD VDD NSLEEPIN HIGH
CHARACTERIZATION_VOLTAGES {0.9}
END_MODE

MODE_NAME S3
PGATE_PAIR TVDD1 VDD NSLEEPIN LOW
CHARACTERIZATION_VOLTAGES {0.9 1.1}
END_MODE

END
```

Data_type: string, optional

`-spice_corners` *corner_list*

Specifies a list of SPICE corners. This is an optional parameter. If you specify SPICE corner(s), the number of corner items should be equal to the number of SPICE model files specified in -spice_models *file_list* .

For example, if a SPICE model1 has three corners and model2 has two corners, you should specify:

```
-spice_models { spice_1.model spice_2.model }
-spice_corners { {s11 s12 s13} {s21 s22} }
```

Based on the above parameter description, the following information gets updated in the library generation file:

```
spice_model_file spice_1.model {s11 s12 s13}
spice_mode_file spice_2.model {s21 s22}
```

Data_type: string, optional

`-spice_models` *`file_list`*

Specifies the names of the SPICE model files used by the SPICE netlist file (`-spice_subckts`).

Some Spice process models use `wnom/Wn` and `lnom/Ln` instead of `w` and `l` (device model parameters names for length and width) for MOS models. Voltus recognizes and supports these alternate parameter names.

Data_type: string, optional

`-spice_netlist_file` *`filename`*

Specifies a file containing all the spice netlists to be used for library generation, the format is one netlist per line. These spice netlist files must contain the subcircuit definitions for the cells being processed by power-grid library generation.

The following example specifies a file called spice_netlist_list_file that contains the list of spice netlist files:

```
set_pg_library_mode -spice_netlist_file spice_netlist_list_file
```

Data_type: string, optional

`-spice_subckts` *`file_list`*

Specifies the names of the SPICE netlist files containing the subcircuits corresponding to the cells being processed during power-grid library generation. These subcircuit netlists will be used to calculate pin capacitances using Voltus library simulator.

Data_type: string, optional

`-spice_subckts_xy_scaling` *`value`*

Specifies the current region scale factor for the Netlist-based flow . The current region scale factor is the factor by which the netlist coordinates are scaled before finding the closest node to attach the tap currents.

The netlist-based flow fails when there is a mismatch in the units of the PGDB coordinates (in nm) and the coordinates in the SPICE netlist causing improper current tap creation. To match these coordinates, you need to specify the current region scale factor. The default scale factor is `1000` considering that the SPICE netlist coordinate unit is `um`.

Data_type: string, optional

`-stop_via` *`layername`*

Stops the extraction of the power-grid network inside a cell at the specified via layer.

Data_type: string, optional

`-stream_files` *`file_list`*

Specifies the names of the GDSII files used during power-grid library generation.

Data_type: string, optional

`-stream_layer_map` *`file`*

Specifies a file that provides the mapping of layers between the LEF and GDS files. The following is a sample GDS layermap file:

```
layer_type    layer_name    gds    gds_layer_no
via             VIA9        gds       59
```

Data_type: string, optional

`-tap_node_distance_file` *`filename`*

Specifies a file containing the distance between tap nodes for the given list of cells. The specified file includes the cell name and tap node distance in micron. The default unit for tap node distance is `micron`, and the default tap node distance is `50 microns`.

This parameter allows you to customize the tap node distance for each cell during power-grid library generation.

File Format:
cell_name tap_node_distance

Examples:

```
OA_cellA 1
OA_cellC 1
```

This parameter is applicable only for standard cell and technology PGV generation (`-cell_type stdcells|techonly`).

Data_type: string, optional

`-tech_stream_layer_map` *filename*

Specifies a file that provides the mapping of the layer names in the Quantus technology file to the layer names in the GDS file.

File Format:

```
technology_layername   use     gds_number  data_type
METAL_1                text      131          0
VIA_1                   x         51          0

METAL_2                text      132          0
VIA_2                   x         52          0
...
```

If `-design_qrc_layer_map` is specified along with `-tech_stream_layer_map`, then the format of `-tech_stream_layer_map` is treated as:

```
lefdef_layername   use     gds_number  data_type
      M1           text       131         0
```

`-temperature` *temp_value_in_degree_celcius*

Specifies the temperature in degree celcius used in library characterization for thermal aware EM/IR analysis.

Default: 25

Data_type: string, optional

`-use_power_switch_data_from_lib {true|false}`

Specifies to read powergate parameters ron, saturation current, and leakage current from the Liberty files specified using the existing `-liberty_files` parameter. This parameter allows you to obtain the powergate parameters from a Liberty file when a Spice netlist is not available, or when the Liberty file contains the powergate parameters.

Data_type: enum, optional

Examples

- The following command sets the power library mode:

```
set_pg_library_mode
-extraction_tech_file RCgen.tch
-lef_layer_map lefdef.map
-cell_type techonly
-power_pins { VDD 1.08 VDDO 1.08 VDDG 1.08 }
-ground_pins { VSS GND VSSG }
-cells_file built_cells_list
-temperature -40
-spice_models models/v1d1_usage.l
-spice_corners { ss_lib }
-spice_subckts { typical_max_m40.spice }
-power_switch_parameters { { cell1 VDDG VDD } { cell2 VDDG VDD } }
-current_distribution propagation
```

- The following command generates a PGV that contains the voltage dependent capacitance table for the specified cells:

```
set_pg_library_mode -cell_type stdcells -cell_list_file cellfile -extraction_tech_file ./../qrcTechFile -lef_layermap
./../lefdef.layermap -temperature 125 -power_pins { vcc 1.26 vccpg 1.26 vccr 1.26 } -ground_pins { vss } -powergate_parameters
{{powercell_pg1 vcc vccpg} {powercell_pg1 vcc vccr} {powercell_pg2 vcc vccpg} {powercell_pg2 vcc vccr}} -spice_corners { tt } -
spice_models spice_model/libspice -spice_subckts { powercell_pg.net powercell_pg2.net } -enable_multi_voltage_cap_generation true
```

  The PGV report includes the following information:

```
INFO: PGV contains voltage dependent cap table for this cell
```

Trigger File Format

For the current characterization flow, you must specify the `-current_distribution { dynamic_simulation` *`trigger_file`* `}` parameter. The format of the trigger file is:

Trigger File Format	Description
#Method of current characterization	
`CURRENT_CHARACTERIZATION_METHOD <method>` `<SIMULATION/DOTLIB/PWL/FSDB/DATASHEET>`	• SIMULATION -> Vector-based flow • DOTLIB -> Liberty based Flow • PWL - User-defined PWL flow • FSDB - Spice total current output captured in the FSDB format • DATASHEET - Datasheet details of any memory
#Memory cell name	
`CELL <`*`cellname1`*`>`	Name of the memory cell
#Mode and conditional details for trigger	
`MODE_NAME <`*`mode1`*`>` `CONDITIONAL_INPUT <conditional input statement for cellname1>`	Mode name and conditional input statement for the cell. Conditional input of current for the given mode.
`CONDITIONAL_PIN <pin_name> (rise|fall|both)`	Conditional pin statement for the cell. Signal pin for which the tool should rise or fall.
`CONDITIONAL_STIMULUS_FILE <usim1.txt>`	Simulation vector detail in the given mode.
`USER_PWL_FILE <pwl1.txt>`	User-specified PWL file that contains information about the PWL waveforms (power/ground currents)
`FSDB_START_TIME <time>` `FSDB_END_TIME <time>`	FSDB Simulation based mode details

`DATASHEET_PARAMETER < {pg_pin1 Capacitance Leakage} {pg_pin2 Capacitance Leakage}>`	Datasheet-based mode details
`END_MODE`	End mode section
#Datasheet clock required parameters	

`DATASHEET_INPUT_SLEW <time>`
`DATASHEET_OUTPUT_SLEW <time>`
`DATASHEET_DELAY <time>`
`DATASHEET_PARAMETER < { power_pin cap leakage_pwr } >`
	• `DATASHEET_INPUT_SLEW` - Clock input slew of the conditional pin
	• `DATASHEET_OUTPUT_SLEW` - Rising slew of the output pin Q
	• `DATASHEET_DELAY` - Mode delay as the read_access and write_access time
	• `DATASHEET_PARAMETER` - Datasheet parameters include power pin voltage, load capacitance, and leakage power. Load capacitance is mode dependent.

#FSDB required parameters	

`FSDB_PATH <path>`
`FSDB_NET_NAME <net> [<generic_name>]`
`FSDB_START_TIME <time>`
`FSDB_END_TIME <time>`
`FSDB_HIER_CHAR <hier_char, for hier DB>`
	• FSDB file path
	• Power net name and its FSDB probe name
	• FSDB start and end time
	• Hierarchical separator for hierarchical database. The default separator is "."
	Single FSDB file having total current for ALL power/ground pins with all mode current in it

#<Other mode independent parameters>	

`DOTLIB_LIST {a list of memory dotlibs files}`
`MODE_COUNT`
`MODE_CONFIG_FILE <file_path>`
`SIM_START_TIME <time>`
`SIM_STOP_TIME <time>`
`SIM_STEP_SIZE <step_size>`
`STIMULUS_FILE <file_path>`
`INCLUDE_FILE <file_path>`
	• `DOTLIB_LIST` - Specifies a list of memory dotlibs.
	• `MODE_COUNT` - Specifies the maximum number of modes to be generated. Valid values are either a positive number or `all'. `all' means stimulus for all power groups are to be generated. The default value is `4`. If you set this parameter to `all`, it may result in high runtime as it may generate too many modes.
	• `MODE_CONFIG_FILE` - Specifies the file containing extra information for memory cell. This file contains the following information - signal slew, signal active value and memory write information.
	• `SIM_START_TIME` - Specifies the simulation start time. The default unit is in nanoseconds (ns).
	• `SIM_STOP_TIME` - Specifies the simulation stop time. The default unit is in nanoseconds (ns).
	• `SIM_STEP_SIZE` - Specifies the step size for power-grid analysis. The default value of step size is 20 picoseconds (ps). This parameter supports other units such as, ns and fs. This parameter supports unit-prefixes (pico, femto, nano) combined with units of measure (ps, fs, ns), but does not support units itself ("second", "Ohm", "Farad") without unit-prefix.
	• `STIMULUS_FILE` - Specifies the stimulus file containing the trigger information for all input ports.
	• `INCLUDE_FILE` - Specifies the name of the simulation include file.

#End cell section	
`END`	End cell section

Current Region File Format

The following shows the current_region_file format:

```
 UNIT CURRENT default_unit_in_amps
UNIT CAPACITANCE default_unit_in_farads
CELL cell_name
<PROP [FLAT | HIER]>
NET net_name
LAYER layer_name
[ REGION x1 y1 x2 y2 current_value <cap_value> | REGION current_value <cap_value> ]
```

UNIT	Defines the default scale to be used with the current and capacitance values read from the current region file. These scales apply to any value that is not followed by an explicit unit specification. Valid current values are: A (Amps), mA (miliAmps), uA (microAmps), and nA (nanoAmps). Valid capacitance values are: F(Farads), mF (miliFarads), uF (microfarads), nF (nanofarads), pF (picoFarads), and fF (femtoFarads).
CELL *cell_name*	This command specifies the name of the cell. All the following NET, LAYER, and REGION commands will apply to this CELL until another CELL specification is encountered. Wildcards can be used in the cell_name.
<PROP [FLAT | HIER]>	If the keyword FLAT is applied, LibGen will not perform hierarchical tracing for the given net in this cell. If the keyword HIER is presented, LibGen will perform hierarchical tracing for the given net in this cell. This provides the ability for LibGen to automatically specify a current value to all instances of a cell in the design hierarchy. The assumption is that all of these instances of the cell will consume the same power. The default is HIER.
NET *net_name*	This command specifies that all the LAYER and REGION commands following it applies to the net_name. This stays in effect until another NET or CELL specification is encountered. Wildcards can be used in the net_name.
LAYER *layer_name*	This command specifies that all the REGION commands following it apply to the layer_name. This stays in effect until another LAYER, NET, or CELL command is encountered. The layer_name must be a library layer name. If the layer is not found, LibGen will issue an error message and exit. You must provide a valid layer name for LibGen to run. If you specified a valid layer, but LibGen does not find any via in the assigned current region, LibGen will issue a warning message, but the process will still continue and exit successfully. In this case, you may get a 0 current value for the specified cell.
REGION *x1 y1 x2 y2 current_value cap_value*	This command specifies the current value and optionally the capacitance value to be associated with the rectangular region specified by the bounding box (*x1 y1 x2 y2*). Current taps will be created at the locations of the vias or contacts specified by the most recent LAYER command. These current taps will be attached to the layer that is below the specified via or contact layer. Coordinate values are assumed to be in GDSII coordinate units and will be interpreted according to the scale and precision settings contained in the GDSII file (or spice netlist with device XY location) that contains the cell. The value of the current drawn by each current tap will be 1/n of the value specified for the current region, where n is the number of taps within the current region. The second variation of the REGION command without the bounding box is valid only if there is only one current region. In this case LibGen can automatically uses the bounding box of the cell for the region. You can specify a single region or multiple regions per cell.

Example of Current Region File

```
UNIT CURRENT uA
UNIT CAPACITANCE fF
CELL an4_80
NET vcc
LAYER CONT
REGION 0 14000 28900 17000 3.0 5.0
NET gnd
LAYER CONT
REGION 1.5e-2mA
```

Example of a Dynamic PWL Region File (Multiple Regions)

```
CELL Cell1
  NET VSS
    PROP HIER
    LAYER via1_r
      REGION 1 10 188 333   { 0 -1 1 9  2 8  3 7  4 6  5 5  6 4  7 3  8 2  9 1  } 1e-12
      REGION 121 10 185 80 { 0 -1 1 90 2 80 3 70 4 60 5 50 6 40 7 30 8 20 9 10 } 4e-12

  NET VDDPR
    PROP HIER
```

```
  LAYER via1_r
    REGION 1 10 188 333 { 0 -1 1 9  2 8  3 7  4 6  5 5  6 4  7 3  8 2  9 1 } 5e-12
NET VDDP
  PROP HIER
  LAYER via1_r
    REGION 1 10 188 333 { 0 -1 1 9  2 8  3 7  4 6  5 5  6 4  7 3  8 2  9 1 } 5e-12
NET VDDAR
  PROP HIER
  LAYER via1_r
    REGION 1 10 188 333 { 0 -1 1 9  2 8  3 7  4 6  5 5  6 4  7 3  8 2  9 1 } 5e-12
...
```

Related Information

- set_advanced_pg_library_mode
- write_pg_library
- rename_pg_library
- "Power-Grid Library Generation" in the *Voltus Stylus Common UI User Guide*

trim_pg_library

```
trim_pg_library
[-help]
[-delete_cells cell1 cell2 ...]
[-delete_cells_file file_name]
-library_list_file file_name
[-out_dir directory_name]
-output_library library_name
```

Specifies to remove a list of cells from the specified PGV. This command allows you to trim the size of a cell library by removing unwanted cells, which can reduce the overall time required for extraction.

Parameters

`-help`	Outputs the command usage.
`-delete_cells cell1 cell2 ...`	
	Default: `None` Specifies the name of a cell or a list of cells to be removed from the power-grid library. *Data_type*: string, optional
`-delete_cells_file file_name`	

	Default: `None`
	Specifies the name of a file containing the list of cells (one per line) to be removed from the power-grid library.
	File Format:
	`cell1`
	`cell2`
	`cell3`
	Data_type: string, optional

`-library_list_file file_name`	

	Default: None
	(**Mandatory**) Specifies the name of a file containing the library or libraries (one per line) that are to be trimmed. If multiple libraries are given, Library Generation merges the multiple same-type power-grid libraries into one consolidated power-grid library.
	File Format:
	`PGV1`
	`PGV2`
	`PGV3`
	Data_type: string, required

`-out_dir directory_name`	
	Default: current working directory Specifies the name of the output directory in which the trimmed power-grid library will be created. *Data_type*: string, optional
`-output_library library_name`	
	Default: None (**Mandatory**) Specifies the name of the trimmed power-grid library created after removing the specified cells. *Data_type*: string, required

Examples

- The following command removes the cell `AND2X2` from the power-grid libraries specified in the `library_list.txt` file and generates one consolidated power-grid library `trimlibrary`:

```
trim_pg_library -library_list_file library_list.txt -output_library trimlibrary -delete_cells AND2X2
```

Related Information

- `merge_pg_library`
- "Power-Grid Library Generation" in the *Voltus User Guide*

write_pg_library

```
write_pg_library
[-help]
[-out_dir dir_name]
[-library_prefix prefix]
```

Generates the output for power-grid library generation.

Parameters

`-help`
Outputs the command usage.
`-library_prefix prefix`
Specifies a prefix for the power-grid library. When this parameter is specified, the default PGV for each cell type is as follows: • techonly - `prefix_techonly.cl (default: techonly.cl)` • stdcells - `prefix_stdcells.cl (default: stdcells.cl)` • macros - `prefix_cellname.cl (default: macros_cellname.cl)` You can generate macro PGVs without the "`macro_`" string in the name and without any prefix by specifying an empty string for `-library_prefix`, as shown below: `write_pg_library -out_dir output_folder -library_prefix ""` *Data_type*: string, optional
`-out_dir dir_name`
Specifies the name of the output directory in which power-grid libraries are created. The default is the current working directory. **Note**: If you specify both `-out_dir output_dir` and `-library_prefix prefix` parameters, the command generates the PGV in the *output_dir* directory with PGV name `prefix_<name>.cl`. *Data_type*: string, optional

Examples

- The following command generates a power-grid library in the *temp* folder:
  ```
  write_pg_library -out_dir temp/
  ```
- The following command specifies to generate a technology library with the default PGV name `techonly` and prefix `ab`:
  ```
  set_pg_library_mode -celltype techonly ...
  write_pg_library -out_dir ./ -library_prefix ab
  ```
  The generated library name will be `ab_techonly.cl`
- The following command specifies to generate a macro library with the default PGV name `<cell_name>` and prefix `ab`:
  ```
  set_pg_library_mode -celltype macros ...
  write_pg_library -out_dir ./ -library_prefix ab
  ```
  The generated library name will be `ab_<cell_name>.cl`

Related Information

- set_pg_library_mode
- set_advanced_pg_library_mode
- write_xpgv
- "Power-Grid Library Generation" in the *Voltus Stylus Common UI User Guide*

write_tech_pg_lib

```
write_tech_pg_lib
[-help]
[-decap_cells cell_list]
[-default_power_voltage value]
[-esd_cells cell_list]
[-filler_cells cell_list]
[-keep {true | false}]
[-lef_layer_map filename]
[-output_dir dir_name]
[-power_switch_file filename]
```

-tech_file *filename*
[-tech_only {true | false}]

Generates technology power-grid library on the fly. The generated tech-PGV will be deleted after the session is over. If this command is specified, the Power Analysis and Rail Analysis engines will automatically use the on the fly generated tech-PGV. The purpose of generating the tech-only PGV on the fly is that you do not need to keep a track of the tech-only PGVs. Thus, it ensures that the tech-only PGV is generated from the correct Quantus technology file.

The tech-only PGV library will be generated under the working directory and will be called by Power/Rail Analysis on the fly, as shown in the following log file snippet:

```
Begin Loading PGV Libraries for Power Calculation
.//pgv/techonly.cl
Ended Loading PGV Libraries for Power Calculation: (cpu:0:00:00,real=0:00:00,mem(process/total/peak)=275.25MB/2585.14MB/1194.52MB)

Begin Processing Cell Libraries for Rail Analysis

Merging libraries:
.//pgv/techonly.cl
../../LIBS/voltus_pgv/macro_pgv/macros_sram.cl
../../LIBS/voltus_pgv/stdcells_pgv/stdcells.cl

Output library: ./work/lib_merged.cl
```

Parameters

`-help`

Outputs the command usage.

`-decap_cells cell_list`

Default: None

Specifies the names of cells that have explicit decoupling capacitance for use in decap optimization. These cells are nearly equivalent to feedthrough cells, except that there is typically an active device between the power and ground nets to provide extra decoupling capacitance. Power-grid view library generation uses this list of cells to ensure that a capacitance value is associated with each cell. If no capacitance is associated with a cell, a warning message will be issued.

`-default_power_voltage value`

Default: 1.2

Specifies voltage for the power pins which are present in LEF.

`-esd_cells cell_list`

Default: None

Specifies a list of Electrostatic Discharge (ESD) cells. You can use this parameter to characterize ESD cells.

When you specify this parameter, the list of ESD cells are later used in the `analyze_esd_network` command while loading the power-grid view library.

`-filler_cells cell_list`

Default: None

List the names of the filler or feedthrough cells that have no GDS data. These cells can be swapped out for decaps during decap optimization.

`-keep {true | false}`

Default: `false`

Specifies that the generated library should be retained after the session completes.

`-lef_layer_map` *filename*

Default: None

Specifies a file that provides the mapping of layers between the LEF and technology file. This parameter is optional. Library Generation automatically generates the layermap file using the technology LEF (containing all layer and via definitions) and Quantus Tech (extraction library) files, therefore, a LEF layermap file is not a mandatory input. The automatically generated layermap file is located in the PGV temp directory.

The following is the format and sample content of the auto-generated file (`lefdefLayerMap_AutoGenerated.txt`):

```
Layer_Type   tech_layername   lefdef    lefdef_layername
via              VIA_9        lefdef      VIA9
metal            METAL_1      lefdef      METAL1
```

You can specify this parameter if you want to provide your own layermap file.

`-output_dir` *dir_name*

Default: current working directory

Specifies the name of the output directory in which tech-only power-grid library is created.

`-power_switch_file` *filename*

Default: None

Specifies a file with the information that technology library generation needs to characterize a powergate cell. Specifies the cell name, unswitched power pin, switched power pin, on resistance in ohms, saturation current in milliamps, and leakage current in milliamps.

File Format:

```
cell supply switched Ron Idsat Ileak
```

Examples:

The following is an example of multiple footer powergate:

```
FOOTBUFVHSV16 VSSG VSS 63 3.167 0.000128252
FOOTBUFVHSV32 VSSG VSS 63 3.167 0.000128252
FOOTBUFVHSV64 VSSG VSS 63 3.167 0.000128252
FOOTBUFVHSV8 VSSG VSS 63 3.167 0.000128252
```

`-tech_file` *filename*

Default: None

(**Mandatory**) Specifies the name of the technology file that will be used for extraction.

`-tech_only {true | false}`

Default: `false`

Specifies to generate the technology PGV without any cells.

Examples

- The following command generates a tech-only power-grid library:

```
write_tech_pg_lib -tech_file $tech_path/qrcTechFile -filler_cells *FILL* -decap_cells *DCAP* -default_power_voltage 0.9
```

write_xpgv

```
write_xpgv
-cell cell_name
[-inst_list inst1 inst2 .. instn ]
[-libgen_include file_name]
[-output_dir directory_name]
[-rail_include file_name]
[-reuse_block_state_directory directory_name]
-state_directory directory_name
[-multi_mode_config config_file]
```

Creates power-grid view (xPGV) models for large digital blocks.The block xPGV can be generated either after the signoff dynamic IR drop analysis or after the dynamic Early Rail Analysis (ERA) of the block. In both cases, the xPGV is generated using the R-C grid of the block and currents captured at the via layer during the analysis. These block-level xPGVs can then be instantiated during the next level of the design or the top-level full-chip analysis. The next level rail analysis will run faster with the block-level xPGVs because these runs use the power-grid that is modeled inside the xPGV.

Parameters

`-help`	Outputs the command usage.
`-cell cell_name`	
	Default: None (**Mandatory**) Specifies the block name for which xPGV model is generated. *Data_type*: string, required
`-inst_list inst1 inst2 .. instn`	
	Default: None Specifies the block instances for which xPGV model is generated. *Data_type*: string, optional
`-libgen_include file_name`	
	Default: None Specifies a file that includes additional library generation options that will be run for xPGV model generation. *Data_type*: string, optional
`-multi_mode_config config_file`	
	Default: None Specifies to generate a single xPGV model with different functional modes. Using this parameter, you can specify a file that includes the different modes and the mapping state directory for each mode. When `-multi_mode_config` is specified, the block-level currents are saved for multiple voltages and frequency (MvMf) per each functional mode in a single PGV. During xPGV instantiation, the different modes are automatically selected based on the power domain to which each mode is connected. The format of the *config_file* file is: *<modeName1> <stateDir_1>* *<modeName2> <stateDir_2>* … *<modeNameN> <stateDir_N>* Before generating MvMf-xPGV, you need to perform multiple rail analysis runs for different corners to generate different state directories (stateDir_*) to be included in the configuration file. *Data_type*: string, required
`-output_dir directory_name`	

	Default: current working directory Specifies the name of the output directory in which the power-grid library report is created. *Data_type*: string, optional
`-rail_include` *file_name*	
	Default: None Specifies a file that includes additional rail analysis options that will be run for xPGV model generation. *Data_type*: string, optional
`-reuse_block_state_directory` *directory_name*	
	Default: None Specifies an existing rail analysis state directory to skip block extraction. The software assumes that the design data (DEFs and GDS) and power-grid libraries are not changed, and will generate xPGV using the extraction data inside the specified state directory. If the state directory does not contain the necessary design and power-grid database files, it will issue an error message and provide information on the missing files. The `-reuse_block_state_directory` parameter is used if the entire design that is analyzed in the rail run should be used to create the xPGV. However, this parameter must not be used in case you require to generate xPGVs for each instance (sub-blocks) of a block. That is, for generating xPGVs for the sub-blocks B and C of the block A (rail analysis was performed on block A), you need to re-extract the block-level grids of B and C as part of xPGV generation for accurate current calculation. In this case, `-reuse_block_state_directory` must not be specified. For net based analysis, you should specify net-based state directory, as shown below: • `write_xpgv -reuse_block_state_directory` *VDD_25C_avg_1* For domain based analysis, you should specify the domain's state directory, as shown below: • `write_xpgv -reuse_block_state_directory` *domain_25C_dynamic_1* A new state directory will be generated for the new analysis with soft links to the old state directory files that were reused. *Data_type*: string, optional
`-state_directory` *directory_name*	
	Default: None (**Mandatory**) Specifies the path to the rail analysis state directory. *Data_type*: string, required

Examples

- The following command creates a xPGV library `hier_block.cl`:
  ```
  write_xpgv -state_directory dynamic_rail/core_125C_dynamic_1/ -cell hier_block -output_dir ../xpgv/
  ```
- The following command creates an xPGV library `super_filter_A` with multiple functional modes specified in the `mode.config` file:
  ```
  write_xpgv -cell super_filter_A -multi_mode_config mode.config -output_dir ./mode_mvmf -rail_include rail.inc
  ```
  The following is a snippet from the mode.config file:
  ```
  label1 dynamic_rail/core_25C_dynamic_1
  lable2 dynamic_rail/core_25C_dynamic_1
  ```

Related Information

- `set_pg_library_mode`
- `set_advanced_pg_library_mode`
- `write_pg_library`
- "Power-Grid Library Generation" in the *Voltus Stylus Common UI User Guide*

28

Power Planning Commands and Attributes

- add_power_mesh_colors
- add_reinforce_pg
- add_reinforce_pg Category Attributes
- add_rings
- add_rings Category Attributes
- add_split_power_vias
- add_stripes
- add_stripes Category Attributes
- add_target_pg
- add_target_pg Category Attributes
- generate_special_via Category Attributes
- trim_pg
- update_power_vias
- route_pg

add_power_mesh_colors

```
add_power_mesh_colors
[-help]
[-reverse_with_non_default_width {0 | 1}]
```

You can use this command to colorize power structures, including power wires and power via metals. This command also fixes color violations of power metals including via metals. It is recommended to use this command once all the power mesh is built.

Note:

1. This command colors only the PG wires and PG via enclosure. but not the PG via cut. Also, it does not color pre-routed signal nets. All signal nets are colored while using the `place_opt_design` or `route_design` command.
2. The command does not fix via cut colors. To fix cut color or trim cut with violation, use the `fix_via` command. This command requires correct node setting in `set_db -design_process_node/-route_design_process_node`.
3. The wide NDR wires will be assigned with a different color than the track color through this command.

Parameter

`-help`	Prints a brief description that includes the type and default information for each `colorizePowerMesh` parameter. For a detailed description of the command and all its parameters, use the `man` command: `man add_power_mesh_colors`
`-reverse_with_non_default_width {0 | 1}`	If set to `1`, this parameter sets the color of PG wire (whose type is not `FOLLOWPIN` and for which the center of wire coincides with a track) to be different with the track color. This parameter only works in the DPT (red and green) layer. *Default*: `0` *Data_type*: enum, optional

Related Topics

Power Planning and Routing chapter in the *Innovus User Guide.*

add_reinforce_pg

```
  add_reinforce_pg
[-help]
[-area {x1 y1 x2 y2}]
[-area_blockage {{x1 y1 x2 y2} {x1 y1 x2 y2 x3 y3 x4 y4 ...} ...}]
[-pattern_file file_name]
[-snap_wire_center_to_grid {default | grid}]
{-pattern_map {{irdrop_value1 pattern_name1} {irdrop_value2 pattern_name2}..} | -write_pattern_template file_name | -
boxes_file file_name| -auto_ir_fix {preroute | postroute}}
[[-tile_row_column {x y} | -tile_size {x y} | -number_region value] | -write_pattern_template file_name]
[-rail_analysis_directory {directory} | -write_pattern_template file_name]
```

Reinforces PG stripes between the existing original PG stripes in Innovus to fix the IR drop violations reported from Voltus.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `add_reinforce_pg` parameter. For a detailed description of the command and all its parameters, use the `man` command: `man add_reinforce_pg`
`-area {x1 y1 x2 y2}`	Specifies the coordinates of the area in which the IR drop violation boxes from Voltus can be fixed by the `add_reinforce_pg` command. *Data_type*: rect, optional

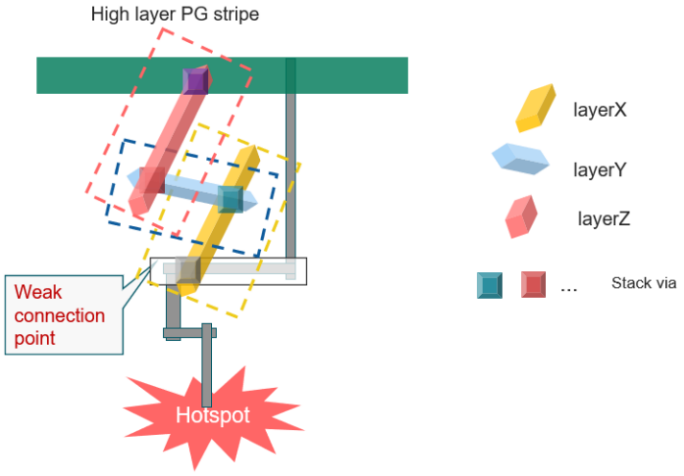
`-area_blockage {{x1 y1 x2 y2} {x1 y1 x2 y2 x3 y3 x4 y4 ...} ...}`	Specifies the blockage coordinates of one or more area regions to restrict the addition of any stripes. This parameter supports both the rectangular and rectilinear area regions along with their combinations. Their spacing is `0` during the stripe generation. Following are the possible parameter usage scenarios:
	• Single Rectangular Area:
	`-area_blockage {x1 y1 x2 y2}`
	For example:
	`-area_blockage {100 100 200 200}`
	• Single Rectilinear Area:
	`-area_blockage {x1 y1 x2 y2 x3 y3 x4 y4 x5 y5 x6 y6}`
	For example:
	`-area_blockage {100 100 100 200 150 200 150 150 200 150 200 100}`
	• Multiple Rectangular Areas:
	`-area_blockage {{x1 y1 x2 y2} {a1 b1 a2 b2} ...}`
	• Multiple Rectilinear Areas:
	`-area_blockage {{x1 y1 x2 y2 ... } {a1 b1 a2 b2 a3 b3 ...} ...}`
	• Combination of Rectangular and Rectilinear Areas:
	`-area_blockage {{x1 y1 x2 y2} {a1 b1 a2 b2} ... {x1 y1 x2 y2 x3 y3 ...} {a1 b1 a2 b2 a3 b3 ...} ...}`
	While blocking multiple areas using this parameter, each of them must be inside the braces. The areas that are not included inside the braces will be ignored. For example, `-area_blockage {{x1 y1 x2 y2} a1 b1 a2 b2 a3 b3 ... {c1 d1 c2 d2 c3 d3 ...}}`, `{x1 y1 x2 y2}`, and `{c1 d1 c2 d2 c3 d3 ...}` will be accepted by the parameter, and `a1 b1 a2 b2 a3 b3 ...` will be ignored.
	Data_type: string, optional

`-auto_ir_fix {preroute | postroute}`	Fixes IR drop with an automated flow. No pattern is needed with the automated flow provided by this parameter. This parameter extracts violation tiles based on RLRP, and automatically fixes the IR drop based on the location provided by Voltus at each layer. When this parameter is used, Voltus rail analysis is performed.

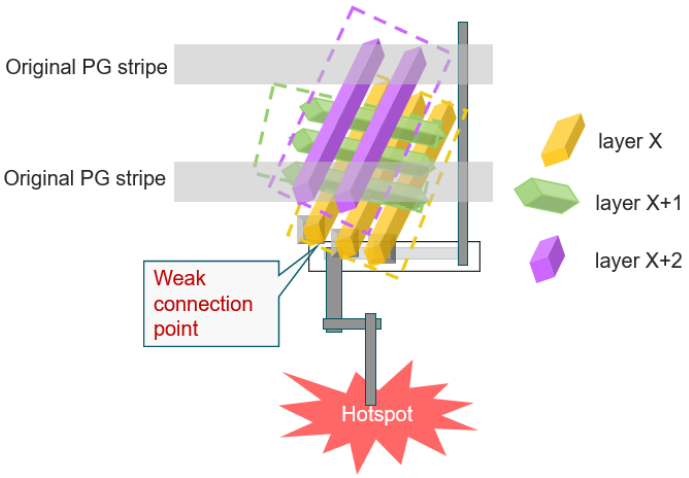
⚠ The tile size and location is provided by Voltus.

Use one of the following values with this parameter:

- `preroute`: Fixes IR drop using minimum resources. Use this flow before signal routing to ensure a good stripe connection. When this parameter value is specified, the software adds several metal layers between the weak point and high PG stripe. The location and layer are automatically controlled by the software analyzer. The following figure presents an example of using `-auto_ir_fix preroute`:



- `postroute`: Fixes IR drop with all the remaining available metal resources. Use this flow after routing so that all the available resources can be used. When this parameter value is specified, the software adds all the rest metal resources at every metal layer according to the information received from Voltus. By this, every metal layer makes connection with the adjacent layer. The layer and tile location are automatically controlled by the software analyzer. By default, `add_reinforce_pg_respect_routes` is set to `all` when `add_reinforce_pg -auto_ir_fix postroute` is specified, which you can change manually. The following figure presents an example of using `-auto_ir_fix postroute`:



⚠ When the `-auto_ir_fix` parameter is used with pattern flow, the pattern is ignored.

Default: `none`

`-boxes_file` *file_name*	Specifies the file for including the list of boxes and corresponding pattern names. The format of the file is mentioned below:
	`Pattern1 x01 y01 x02 y02`
	`Pattern2 x11 y11 x12 y12`
	……
	Data_type: string, optional

`-number_region` *value*	Specifies the number of regions to be analyzed by the software. The integer value of this parameter ranges from 1 to 65536. *Default:* `100` *Data_type*: int, optional
`-pattern_map {{`*irdrop_value1 pattern_name1*`} {`*irdrop_value2 pattern_name2*`}..}`	
	Specifies the pattern for IR drop fixing through the IR drop value and pattern name defined by the `-pattern_file` parameter. If `irdrop_value1` is lesser than `irdrop_value2`, the `add_reinforce_pg` command uses `pattern_name1` to fix the IR drop between `irdrop_value1` and `irdrop_value2`. And `pattern_name2` is used to fix the IR drop bigger than or equal to `irdrop_value2`. It is recommended to specify the `irdrop_value` in the ascending order. *Data_type*: string, optional
`-rail_analysis_directory {`*directory*`}`	Allows the `add_reinforce_pg` command to read the rail analysis information available in the specified directory. *Data_type*: string, optional
`-snap_wire_center_to_grid {default | grid}`	Specifies the snap wire center for routing grid. Use this parameter to control the added stripe for turning on or off the grid for the `add_reinforce_pg` command. Following are the parameter values: • `default`: Snaps the added wires to gird or half grid, according to the wire width. • `grid`: Snaps the the added wires only to the grid. *Default:* *`default`* *Data_type*: string, optional
`-tile_row_column {x y}`	Specifies the number of tiles for power analysis in the x and y directions. *Data_type*: string, optional
`-tile_size {`*x y*`}`	Specifies the size of tiles for power analysis in the x and y directions. This parameter enables tile-based analysis of the hot-spot regions to divide the design based on user-defined tile size and reports IR drop violations on each tile. *Data_type*: string, optional
`-write_pattern_template` *file_name*	Specifies the template of the pattern file. This parameter cannot be used with other parameters. Refer to the following example to understand the usage of this parameter: `add_reinforce_pg -write_pattern_template pattern.cfg` *Data_type*: string, optional

`-pattern_file` *file_name*	Defines the pattern for the pattern template of reinforce stripes. The `add_reinforce_pg` command uses different patterns for fixing the corresponding IR drop value sections. The following is the file syntax of the pattern template for single metal layer:
	`NAME  pattern_name`
	`NETS  {list_of_nets}`
	`LAYER  metal_layer`

```
MAXPITCH         real_value
MINDISTANCE         real_value
WIDTH  real_value
VIACONTROL      {metal_of_viaHookUpBot metal_of_viaHookUpTop}
SPACING  real_value                              ##optional
UDA  subclass_string                      ##optional
```

Note: You can define the density of the added PG using the MAXPITCH and MINDISTANCE parameters.

The following is the description of the pattern template parameters for a single metal layer:

- NAME: Specifies the name of the pattern. The pattern_name used at pattern_map should same as the name defined in the pattern file.
- NETS: Specifies the names of the nets for which the stripes are required to be generated.
- LAYER: Specifies the name of the layer in which the stripes are required to be generated.
- WIDTH: Specifies the width of the stripes.
- VIACONTROL: Specifies the metal layers to which the generated PG wires will be connected.
- SPACING: Specifies the spacing between the stripes/nets defined for the NETS attribute. If SPACING is not defined, the Multi nets defined through NETS will use the minimum DRC free distance.
- UDA: Displays the user-defined attribute attached to every generated stripe segment.
- MAXPITCH: Specifies the maximum distance between the generated PG stripes. The distance between the added PG wires and original PG stripes cannot be more than MAXPITCH.
- MINDISTANCE: Specifies the minimum distance between the generated PG stripes. The distance between added PG wires and original PG stripes should be more than MINDISTANCE.

Else than the above parameters, you can use an optional keyword, OVERVIAMETAL *metal_layer*, which makes the add_reinforce_pg command create PG over the via metal of same PG net. Else than the OVERVIAMETAL keyword, the *metal_layer* parameter is also optional. If *metal_layer* is defined with the OVERVIAMETAL keyword, only the defined metal layers follow the OVERVIAMETAL rule. Else, all the metal layers defined at the LAYER need to follow the OVERVIAMETAL rule.

Note: You can define the density of the added PG using the MAXPITCH and MINDISTANCE parameters. If the distance between the two PG wires for any net is smaller than MINDISTANCE after adding the PG, the added PG wires will be deleted.

The following is the file syntax of pattern template for two metal layers:

```
NAME  pattern_name
NETS    {list_of_nets}
LAYER  metal_layer1 metal_layer2
MAXPITCH         real_value1   [real_value2]
MINDISTANCE         real_value1   [real_value2]
WIDTH  real_value1   [real_value2]
VIACONTROL      {metal_of_viaHookUpBot metal_of_viaHookUpTop} [{metal_of_viaHookUpBot
metal_of_viaHookUpTop}]
SPACING  real_value                              ##optional
UDA  subclass_string                    ##optional
```

Note: If both the MAXPITCH and MINDISTANCE parameters define the same value and the two added metal layers are in the same direction, then metal_layer2 should align with metal_layer1.

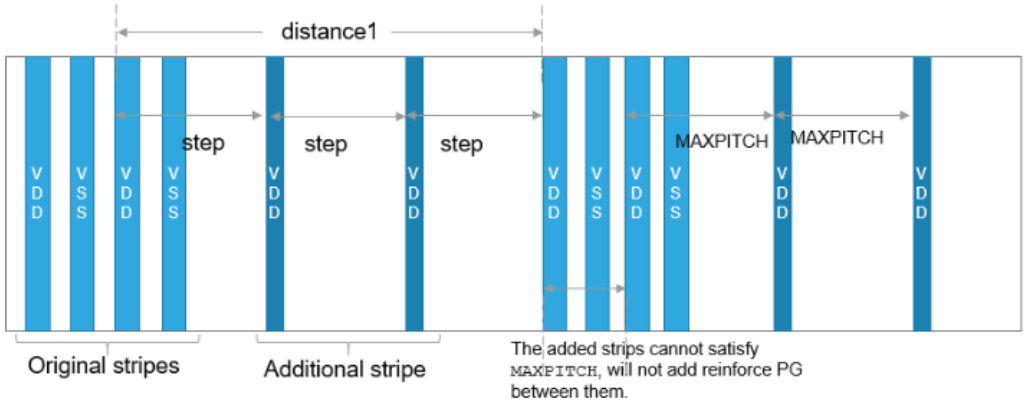
The following is the description of the pattern template parameters for two metal layers:

- NETS: Specifies the net names of the two metal layers to be created.
  Note: The list of nets is not based on the metal layers. Every metal layer needs to generate all the nets defined.
- LAYER: Specifies the name of the layers in which stripes are to be created. Two metal layers are required in this scenario.
- WIDTH, VIACONTROL, SPACING, MAXPITCH, and MINDISTANCE: These parameters can be defined twice for

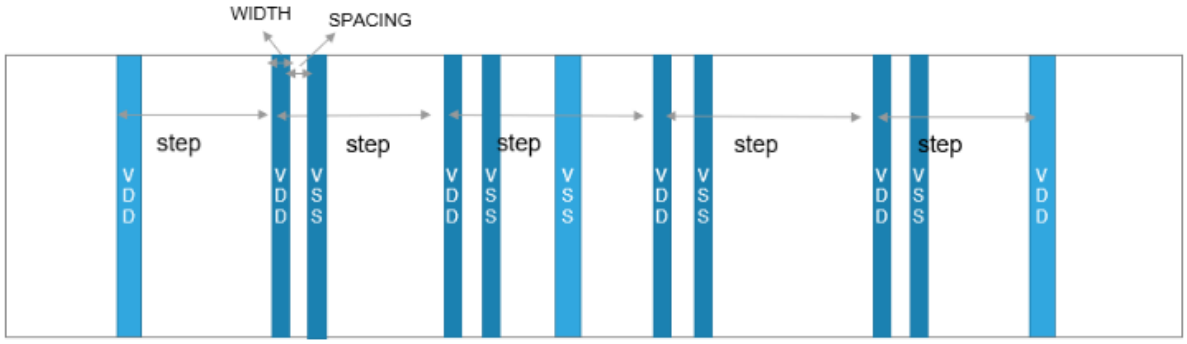
	the corresponding metal layer. If only one value is specified for these parameters, two metal layers will use the same value. You can use an optional keyword, `OVERVIAMETAL metal_layer`, which makes the `add_reinforce_pg` command create PG over the via metal of same PG net. Else than the `OVERVIAMETAL` keyword, the *`metal_layer`* parameter is also optional. If *`metal_layer`* is defined with the `OVERVIAMETAL` keyword, only the defined metal layers follow the `OVERVIAMETAL` rule. Else, all the metal layers defined at the `LAYER` need to follow the `OVERVIAMETAL` rule. The following is the syntax of the `OVERVIAMETAL` keyword: `OVERVIAMETAL metal_layer*`

Examples

- The following command presents an example of defining a pattern file parameters and calculating the number of additional stripes to be added between two adjacent stripes of the same net:
  ```
  add_reinforce_pg -pattern_file pattern.tcl -pattern_map {0.03 PATTERN1}
  ```
  The following is the output of the above command:

  The following are the contents of the defined `pattern.tcl` file:
  ```
  NAME  PATTERN1
  NETS   VDD
  LAYER  M5
  MAXPITCH    15
  MINDISTANCE      5
  WIDTH  0.3
  VIACONTROL  {M4 M6}
  UDA  PGFIX_1
  ```
  Additional stripes can only be added when the distance between the two adjacent stripes of the same net is more than `MAXPITCH` and `2*MINDISTANCE`, as shown below:
  ```
  Distance1 > MAXPITCH  &&  Distance1 > 2*MINDISTANCE
  ```
  To calculate the number of additional stripes to be added between two adjacent stripes of the same net:
  ```
  N = ceil (distance1 / MAXPITCH)
  ```
  The `N-1` stripes will be evenly distributed between two adjacent stripes with:
  ```
  step = distance1/N
  ```
  Note: When some new stripes added between two adjacent stripes are shifted and the distance between them is more than the `Maxpitch`, the tool adds extra stripe to satisfy the `Maxpitch` defined.
- The following command defines a metal layer with two nets:
  ```
  add_reinforce_pg -pattern_file pattern.tcl -pattern_map {0.06 pattern2}
  ```
  The following is the output of the above command:



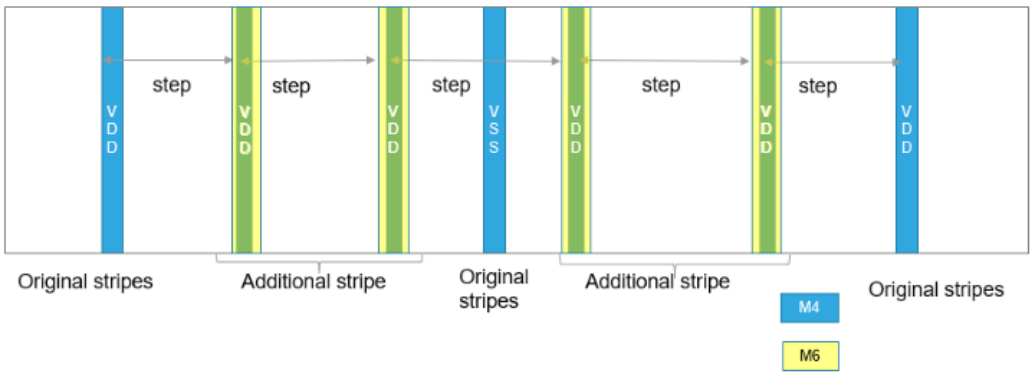
The following are the contents of the defined `pattern.tcl` file:

```
NAME        pattern2
NETS         VDD VSS
LAYER                M5
MAXPITCH          10
MINDISTANCE    3
WIDTH         0.3
SPACING         0.3
VIACONTROL     {M2 M8}
UDA         PGFIX_2
```

- The following command defines two metal layers in the same direction and the `MAXPITCH` and `MINDISTANCE` parameters with one value. Here, the `M6` metal layer should align with `M4`. The number and location of the additional stripes will depend on the first layer defined:

  ```
  add_reinforce_pg -pattern_file pattern.tcl -pattern_map {0.06 pattern3}
  ```

  The following is the output of the above command:



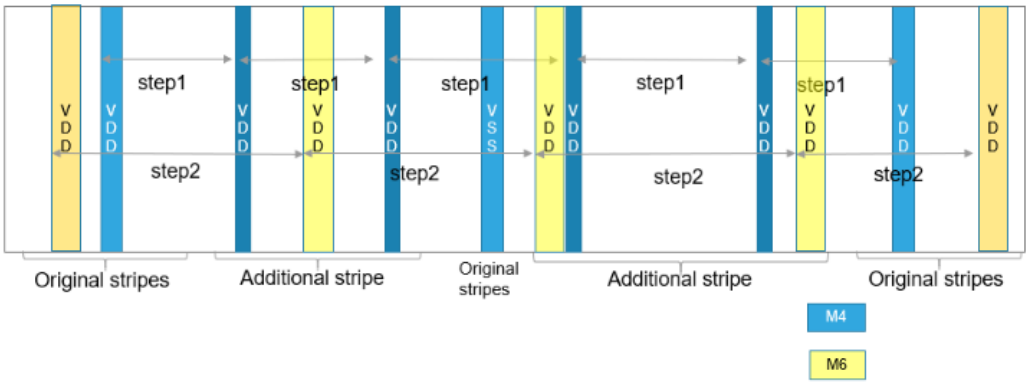
  The following are the contents of the defined `pattern.tcl` file:

  ```
  NAME  pattern3
  NETS   VDD
  LAYER  M4 M6
  MAXPITCH       20
  MINDISTANCE    5
  WIDTH  0.3  0.5
  VIACONTROL {M3 M5}  {M5 M7}
  UDA  PGFIX_3
  ```

- The following command defines two metal layers with same direction and and `MAXPITCH` and `MINDISTANCE` parameters with two values. The additional stripes of two metal layers will also be added respectively.:

  ```
  add_reinforce_pg -pattern_file pattern.tcl -pattern_map {0.06 pattern4}
  ```

  The following is the output of the above command:



The following are the contents of the defined `pattern.tcl` file:

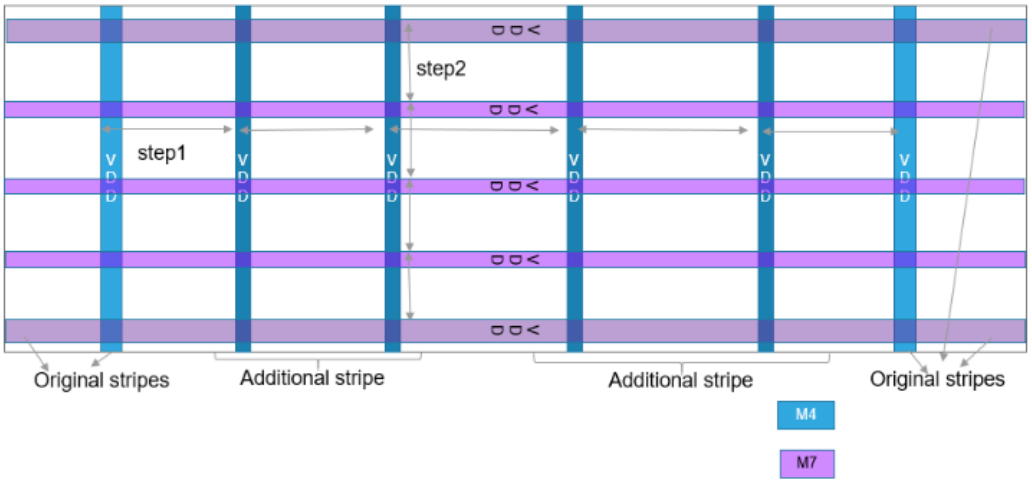
```
NAME  pattern4
NETS   VDD
LAYER  M4 M6
MAXPITCH       20 30
MINDISTANCE    5   8
WIDTH  0.3  0.5
VIACONTROL {M3 M5}  {M5 M7}
UDA  PGFIX_3
```

The `M4` and `M6` metal layers calculate this step respectively, where `M4` uses `step1` and `M6` uses `step2` to generate PG.

- The following command defines two metal layers with different directions and `MAXPITCH` and `MINDISTANCE` parameters with two values. The additional stripes of two metal layers will also be added respectively.:

```
add_reinforce_pg -pattern_file pattern.tcl -pattern_map {0.06 pattern5}
```

The following is the output of the above command:



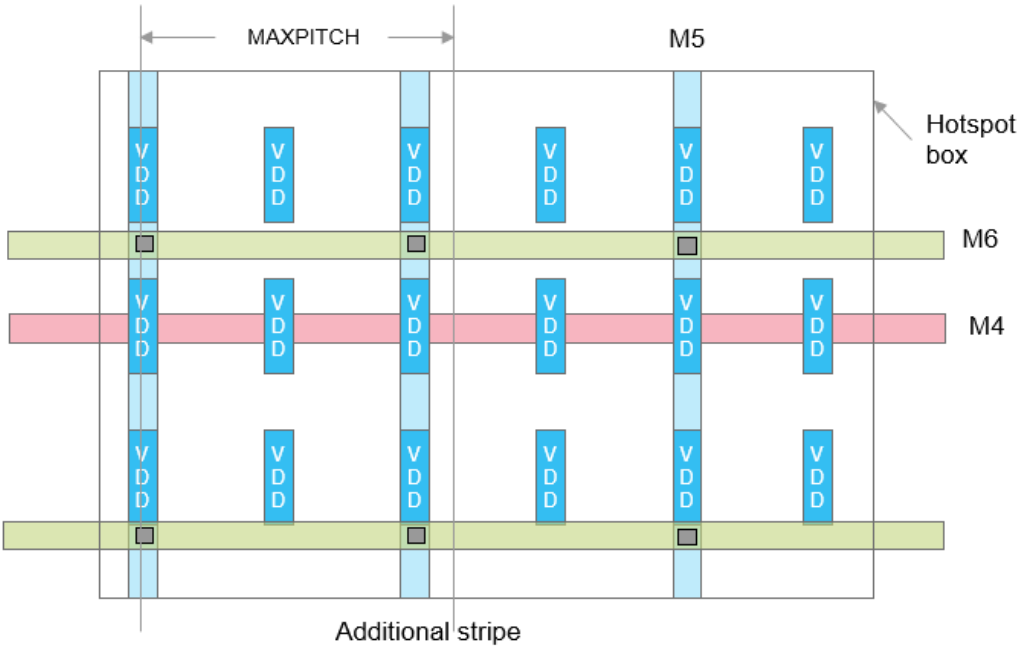
The following are the contents of the defined `pattern.tcl` file:

```
NAME  pattern5
NETS   VDD
LAYER  M4 M7
MAXPITCH       20 30
MINDISTANCE    5   8
WIDTH  0.3  0.5
VIACONTROL {M1 M5}  {M6 M10}
UDA  PGFIX_3
```

- The following command defines a metal layer with the `OVERVIAMETAL` keyword:

  ```
  add_reinforce_pg -pattern_file pattern.tcl -pattern_map {0.06 pattern6}
  ```

  The following is the output of the above command:



  The following are the contents of the defined `pattern.tcl` file:

  ```
  NAME pattern6
  OVERVIAMETAL
  NETS VDD
  LAYER M5
  MAXPITCH    15
  MINDISTANCE      5
  WIDTH 0.3
  VIACONTROL  {M4 M6}
  UDA PGFIX_1
  ```

Related Topics

Power Planning and Routing chapter in the *Innovus User Guide.*

add_reinforce_pg Category Attributes

The root attributes in the `add_reinforce_pg` category are:

- add_reinforce_pg_auto_ir_fix_effort
- add_reinforce_pg_critical_path_slack
- add_reinforce_pg_irdrop_hierarchical_block
- add_reinforce_pg_irdrop_hierarchical_top_rail_analysis_directory
- add_reinforce_pg_respect_defined_nets
- add_reinforce_pg_respect_routes
- add_reinforce_pg_respect_stdcell_geometry
- add_reinforce_pg_timing_aware_effort

To set the value of any root attribute, use the `set_db` command:

```
set_db attribute_name value
```

To get the current value of any root attribute, use the `get_db` command:

```
get_db attribute_name
```

To get the current value of all attributes in the `add_reinforce_pg` category, use the following command:

```
get_db -category add_reinforce_pg
```

add_reinforce_pg_auto_ir_fix_effort

```
add_reinforce_pg_auto_ir_fix_effort {low | medium | high}
```

Default: `low`

Data_type: enum, read/write

Specifies the IR fixing effort level during the stripe generation. Use this parameter to control the IR fixing effort. The number of fixed tiles is changed according to the effort settings. The higher the effort value, the larger number of tiles are passed back from Voltus to RPG for polygon insertion, which results in insertion of more polygons.

⚠ This parameter works only with the `add_reinforce_pg -auto_ir_fix` parameter.

add_reinforce_pg_critical_path_slack

```
add_reinforce_pg_critical_path_slack slack
```

Default: `0`

Data_type: int, read/write

Specifies a slack threshold value (in nanoseconds) for critical nets during timing-driven PG stripe reinforcement. It means that `add_reinforce_pg` honors all the violation paths with slack value lesser than the specified `add_reinforce_pg_critical_path_slack` value, since all such associated nets are considered critical.

add_reinforce_pg_irdrop_hierarchical_block

```
add_reinforce_pg_irdrop_hierarchical_block block
```

Default: `""`

Data_type: string, read/write

Reports the IR drop violations on the specified list of hierarchical block(s) to fix IR drop. The names of the specified hierarchical blocks should be instantiations of the same master block, which is the loaded block. If you specify the `*` value of this parameter, all the instantiations of the loaded

block are considered.

add_reinforce_pg_irdrop_hierarchical_top_rail_analysis_directory

`add_reinforce_pg_irdrop_hierarchical_top_rail_analysis_directory {directory_name}`

Default: ""

Data_type: string, read/write

Reads the rail analysis information of full chip available in the specified directory for the hierarchical flow blocks. It collects the IR drop information of the specified block and uses it to fix the IR drop.

add_reinforce_pg_respect_defined_nets

`add_reinforce_pg_respect_defined_nets {file_name}`

Default: {}

Data_type: string, read/write

When this attribute is used, `add_reinforce_pg` respects the user-defined nets listed in the specified file, which are critical in stripe generation. This helps in reserving space around the specified nets, and the newly added PG wires are DRC clean with these nets. This attribute is independent of the `add_reinforce_pg_respect_routes` and `add_reinforce_pg_respect_critical_path` attributes. However, when it is used together with the `add_reinforce_pg_respect_routes` and `add_reinforce_pg_respect_critical_path` attributes, `add_reinforce_pg` respects all of them.

add_reinforce_pg_respect_routes

`add_reinforce_pg_respect_routes {none | fixed | fixed_and_clock | all}`

Default: `fixed_and_clock`

Data_type: enum, read/write

Specifies the route types to honor check of pre-existing routes during stripe generation. This attribute always honor all covers and PGs, but does not honor the trail routes, Following are the attribute values:

- `none`: Specifies that no route types are respected.
- `fixed`: Respects fixed regular wires.
- `fixed_and_clock`: Respects fixed and routed clocks.
- `all`: Respects fixed and routed clocks along with the routed signals.

The following table presents the support of the `add_reinforce_pg_respect_routes` attribute values with various statuses of net:

Target PG Honoring a Wire			**add_reinforce_pg_respect_routes Values**			
Net Use	**Status**		**none**	**fixed**	**fixed_and_clock**	**all**
P/G			Yes	Yes	Yes	Yes
Non-P/G	**Cover**		Yes	Yes	Yes	Yes
	Fixed		No	Yes	Yes	Yes
	Routed	**Clock**	No	No	Yes	Yes
		Signal	No	No	No	Yes
	Unknown		No	No	No	No

add_reinforce_pg_respect_stdcell_geometry

`add_reinforce_pg_respect_stdcell_geometry {none sequential all}`

Default: `all`

Data_type: enum, read/write

Specifies the cell pin types to be respected in via generation. This attribute ignores the standard cell pins to control DRC violation checking on standard cell pins in via generation.

add_reinforce_pg_timing_aware_effort

`add_reinforce_pg_timing_aware_effort {none | low | medium | high}`

Default: `none`

Data_type: enum, read/write

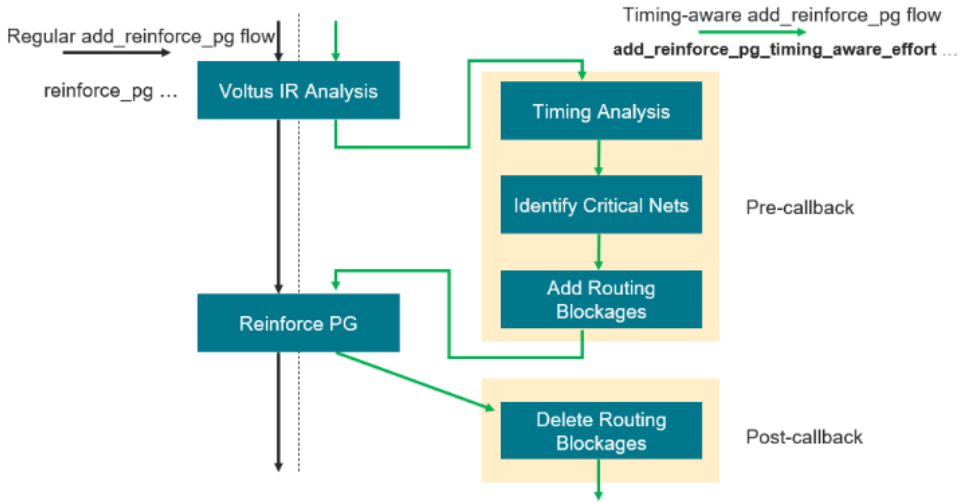
Specifies the timing aware effort level on critical paths of `add_reinforce_pg`. The `add_reinforce_pg` command performs timing analysis, when the `add_reinforce_pg_timing_aware_effort` attribute is set to `low`, `medium`, or `high`. Use the `add_reinforce_pg_critical_path_slack` attribute to determine the slack range.

When the `add_reinforce_pg_critical_path_slack` attribute is not defined, following will be the behavior of the non-default values of `add_reinforce_pg_timing_aware_effort`:

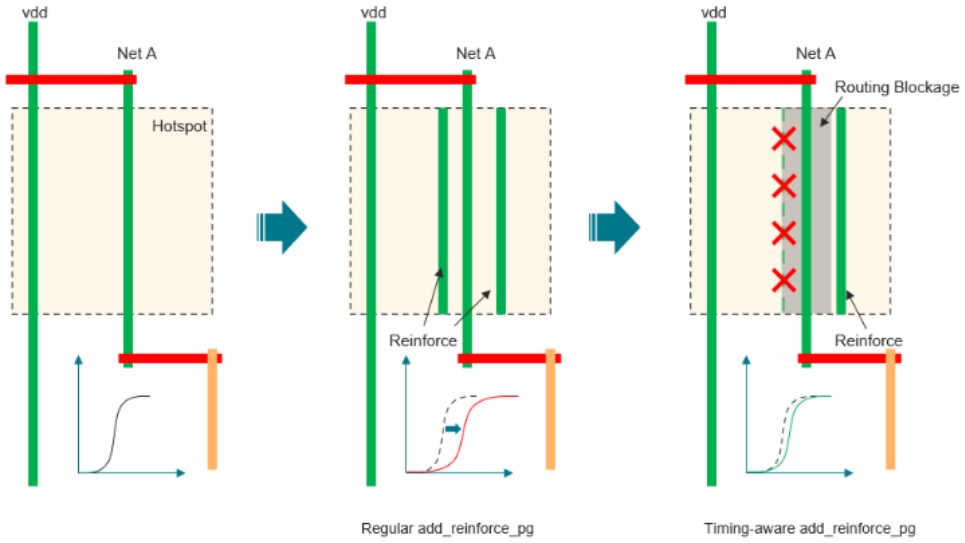
- `low`: Timing clock protection is performed, which uses slack value and reserve space size.
- `medium`: Performs clock protection, with `max_slack` as 10+ ps (since 10ps is used to cover the borderline paths), `max_path` as `number_of_flops`, and `nworst` as 1. You can set the `add_reinforce_pg_critical_path_slack` attribute to overwrite this value, `max_path` to cover all the end points and single worst path per end point.
- `high`: Performs clock protection, with `max_slack` as 10+ ps (since 10ps is used to cover the borderline paths), `max_path` as `10 * number_of_flops`, and `nworst` as `10,2 * blockage_size`. You can set the `add_reinforce_pg_critical_path_slack` attribute to overwrite this value, `max_path` to cover all the end points and single worst path per end point, with covering 10 worst paths per end point.

⚠ The `add_reinforce_pg_critical_path_slack` attribute can work only when the `add_reinforce_pg_timing_aware_effort` attribute is set to `low`, `medium`, or `high`.

See the figure below for understanding the timing aware reinforce PG flow:



The following figure presents reserving of spaces around critical net using with and without the `add_reinforce_pg_timing_aware_effort` attribute:



add_rings

```
add_rings
[-help]
[-around {each_block each_reef power_domain default_power_domain selected cluster shared_cluster user_defined}]
[-center {0 | 1}]
[-exclude_selected {0 | 1}]
[-extend_corners {tl tr bl br lt lb rt rb}]
[-follow {core io}]
[-jog_distance real_value]
[-nets {name-or-value}]
[-offset_adjustment {automatic fixed}]
[-rectangle {0 | 1}]
[-skip_side {top bottom left right}]
[-snap_wire_center_to_grid {none grid half_grid either}]
[-threshold {value | auto}]
[-type {core_rings block_rings}]
[-user_class subclass_string]
[-use_interleaving_wire_group {0 | 1}]
[-use_wire_group {0 | 1}]
[-use_wire_group_bits value]
[-use_wire_group_reinforcement {0 | 1}]
[-use_wire_group_reinforcement_group_via {0 | 1}]
[-use_wire_group_reinforcement_spacing_width {spacing width}]
[-user_defined_region <x1 y1 x2 y2 ...>]
{-layer {layer | {top top_layer bottom bottom_layer left left_layer right right_layer}} } {-width {value | {top top_width bottom
bottom_width left left_width right right_width}} } {-spacing {value | {top top_spacing bottom bottom_spacing left left_spacing
right right_spacing}} }
[-offset {value | {top top_offset bottom bottom_offset left left_offset right right_offset}}]
```

Creates rings for specified nets around the core boundary or selected blocks and groups of core rows. Use this command after creating an initial floorplan.

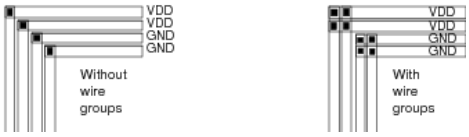
Parameters

`-help`	Outputs a brief description that includes the type and default information for each `add_rings` parameter. For a detailed description of the command and all its parameters, use the `man` command: `man add_rings`
`-around`	Specifies the objects around which to create rings. This parameter is used in conjunction with the `-type` parameter. Specify one of the following: • `each_block|each_reef|power_domain`: Creates rings around each power domain or around each fence that surrounds a soft block in the design. This option can only be used if you also specify `-type block_rings`. • `default_power_domain`: Generates ring around a default power domain. • `selected|cluster|shared_cluster`: Creates rings around all selected blocks with shared edges residing in the middle of the common areas between adjacent blocks. This option can only be used if you also specify `-type block_rings`. • `user_defined`: Creates rings around user-defined blocks. *Data_type*: enum, optional
`-center {0 | 1}`	Specifies whether to center the core rings between the I/O pads and core boundaries. If you do not specify this parameter with a value of 1, you must specify the parameter `-offset {value | {top top_offset bottom bottom_offset left left_offset right right_offset}`. *Default:* `0` *Data_type*: bool, optional
`-exclude_selected {0 | 1}`	Specifies whether the core ring should be created so that selected objects (blocks and power domains) are not included within its boundary. *Default:* `0` *Data_type*: bool, optional
`-extend_corners {[tl] [tr] [bl] [bt] [lt] [lb] [rt] [rb]}`	
	Extends the specified segment of the ring to the specified direction. Following are the parameter values: • `bl`: Extends the bottom segment of the ring to the left until encountering a target. • `br`: Extends the bottom segment of the ring to the right until encountering a target. • `lb`: Extends the left segment of the ring to the bottom until encountering a target. • `lt`: Extends the left segment of the ring to the top until encountering a target. • `rb`: Extends the right segment of the ring to the bottom until encountering a target. • `rt`: Extends the right segment of the ring to the top until encountering a target. • `tl`: Extends the top segment of the ring to the left until encountering a target. • `tr`: Extends the top segment of the ring to the right until encountering a target. *Data_type*: (enum)+, optional
`-follow {core|io}`	Specifies whether core rings are placed along the core boundary or the I/O boundary. This parameter is used only with the `-type core_rings` parameter. *Default:* `core` *Data_type*: enum, optional

`-jog_distance` *value*	Specifies the least amount of jog allowed (to follow the contour of the referenced object) before a jog is removed. This value is in micrometers. If the ring would need to jog a distance equal to or less than this value in order to follow the contour of the object, the ring will not jog. *Default:* If you do not specify this parameter, the jog distance value is automatically calculated based on the worst spacing in the technology file. *Data_type*: string, optional
`-layer {layer | {top top_layer bottom bottom_layer left left_layer right right_layer}}`	
	Can now be used to replace old options: `-layer_top / - layer_bottom / - layer_left / -layer_right` `-layer metal7` means all four sides of the ring will use layer metal7. **Example:** • `-layer metal7` : all four sides of the ring will use layer metal7. • `-layer {top metal7 bottom metal7 left metal8 right metal8}`: top and bottom ring wires will be generated on metal7, left and right ring wires will be on metal8. The following types of formulas are supported: 1. Lefname 2. Metal layer number **Example:** If Metal2 is named by 'MET2' in LEF file, then you can use 1. MET2 2. 2 to identify metal2. *Data_type*: string, required
`-nets` *list_of_nets*	Specifies the names of the nets for which power rings are to be created. You must enclose the list of net names within curly braces. *Data_type*: string, required
`-offset_adjustment` *automatic* | *fixed*	Specifies whether the offset is automatically adjusted when you change the width or spacing values. *Default:* `automatic` *Data_type*: enum, optional
`-offset {value | {top top_offset bottom bottom_offset left left_offset right right_offset}}`	

	Can now be used to replace old options:
	`-offset_top / -offset_bottom / -offset_left /`
	`-offset_right`
	`-offset 5` means all four sides of the ring will use 5um offset.
	Example:
	• `-offset metal7`: 5 means all four sides of the ring will use 5um offset. (existing)
	• `-offset {top 5 bottom 6 left 7 right 8}` : top ring wires will use 5um offset, bottom ring wires will use 6 um offset, left ring wires will use 7 um offset, right ring wires will use 8 um offset.
	Data_type: string, optional

`-rectangle {0 | 1}`	If set to `1`, creates rectangular rings only. *Default:* If you do not specify this parameter or if you specify it with a value of `0`, the creation of rectilinear rings is permitted. *Data_type*: bool, optional
`-skip_side`	Can now be used to replace old options: `-top / -bottom / -left / -right` `-skip_side {left}` has is the same as `-left 0` in the old behavior. *Data_type*: (enum)+, optional
`-snap_wire_center_to_grid none | grid | half_grid | either`	
	Controls the snapping of the center of the rings. If you do not specify this parameter, rings are not snapped to the routing grid. Specify one of the following values: Does no snapping of the wires. Snaps the centers of the wires to the routing grid of the same layer as the wire. Snaps the centers of the wires to the half-routing grid. The half-routing grid is a virtual grid with the tracks at the center of every two adjacent tracks of the routing grid. Snaps the centers of the wires to the closer of the routing grid or half-grid track. *Data_type*: enum, optional
`-spacing spacing`	Can now be used to replace old options: `-spacing _top / -spacing _bottom / -spacing _left / -spacing _right` `-spacing 1` means all four sides of the ring will use 1um for the spacing. Example: • `-spacing 1` : all four sides of the ring will use 1um for the spacing. (existing) • `-spacing {top 1 bottom 1 left 2 right 2}` : top ring wires will use 1um for the spacing, bottom ring wires will use 1 um for the spacing, left ring wires will use 2 um for the spacing, right ring wires will use 2 um for the spacing. *Data_type*: string, optional
`-threshold {value|auto}`	• `value`: Specifies the least amount of spacing allowed between ring segments of adjacent blocks before the rings are merged. If the spacing is less than this value, the common sides of the rings are merged. Units in micrometers. • `auto`: Specifies that the Innovus will automatically determine where to merge the wires being created with the pre-routed wires. *Default:* 10 µm (if you do not specify this parameter, rings farther than 10 µm from each other are not merged). *Data_type*: string, optional

`-type` *core_rings	block_rings*	Specifies the ring type to create. *Default:* If you do not specify this parameter, core rings are created. Specify one of the following values: Creates core rings that follow the contour of the core boundary or the I/O boundary. If you specify `core_rings`, then you must specify either the `-follow` parameter or the `-around user_defined` parameter. *Default:* The ring is offset from the core, based on offset values that you specify in the `-offset` parameters. `block_rings` Creates block rings that follow the contour of the specified objects. If you specify `block_rings`, then you must specify the `-around` argument with one of its required options. *Data_type*: enum, optional
`-user_class` *subclass_string*	Displays the user defined attribute attached to each created ring segment. *Data_type*: string, optional
`-user_defined_region {`*x1 y1 x2 ya …*`}`	Specifies the coordinates for a rectilinear region if you use the `-around user_defined` parameter. You must specify at least four sets of coordinates and the coordinate list must be enclosed within curly braces. You must also specify the coordinates in a linear sequence. For example, you cannot specify the coordinates in a sequence such as bottom left, top right, bottom right, top left. *Data_type*: string, optional
`-use_interleaving_wire_group {0 | 1}`	If set to 1, connects multiple wires from the same net together, even if separated by wires from a different net. You can only specify this parameter if you also specify `-use_wire_group 1`. *Default:* If you do not specify this parameter or if you specify it with a value of `0`, interleaving wire groups are not used. *Data_type*: bool, optional
`-use_wire_group {0 | 1}`	If set to `1`, connects multiple wires from the same net together. **Note:** For stripes to connect to rings using wire groups, you must specify `-use_wire_group` in both the `add_rings` and `add_stripes` commands. The following figure shows the difference in the connections.  *Default:* If you do not specify this parameter or if you specify it with a value of `0`, wire groups are not used. *Data_type*: bool, optional
`-use_wire_group_bits` *integer*	Specifies how many nets are to be included in the wire group. You can only specify this parameter if you also specify `-use_wire_group 1`. *Default:* The same number of nets as specified in the `-nets` parameter. *Data_type*: int, optional
`-use_wire_group_reinforcement {0 | 1}`	Specifies reinforcement stripes. The option is boolean and optional. *Default:* 0 *Data_type*: bool, optional

`-use_wire_group_reinforcement_group_via {0 | 1}`	Creates group vias for wire group corners along with reinforcement stripes placed in the horizontal and vertical directions for the same wires. You can use this parameter to improve the yield and reliability issues associated with vias and reinforcement stripes by ensuring that if one of the vias or reinforcement stripes produces a void, the contact will be maintained through one of the adjoining vias or reinforcement stripes. *Default:* `0` *Data_type*: bool, optional
`-use_wire_group_reinforcement_spacing_width` *spacing width*	
	Creates group vias for wire group corners along with reinforcement stripes placed in the spacing and width for the same wires. You can use this parameter to improve the yield and reliability issues associated with vias and reinforcement stripes by ensuring that if one of the vias or reinforcement stripes produces a void, the contact will be maintained through one of the adjoining vias or reinforcement stripes. *Data_type*: string, optional
`-width` *width*	Can now be used to replace old options: `-width_top / -width_bottom / -width_left / -width_right` `-width 4` means that four sides of the ring will use 4um metal width. Example: • `-width 4` : all four sides of the ring will use 4um metal width. (existing) • `-width {top 4 bottom 4 left 6 right 6}` : top ring wires will use 4um metal width, bottom ring wires will use 4um metal width, left ring wires will use 6um metal width, right ring wires will use 6um metal width. *Data_type*: string, required

Examples

- The following command adds a core ring to the `vdd` and `gnd` nets on layers `METAL5` and `METAL6`:

  ```
  add_rings -nets {vdd gnd} -type core_rings -center 1 -layer {top METAL6 bottom METAL6 right METAL5 left METAL5} -width 10 -spacing 2
  ```

  This ring is centered between the I/O pad area and the core boundary, has a width of 10 µm, and is located at least 2 µm away from any other ring.

- A user may use `add_rings -nets {VSS VDD} -layer {top metal5 bottom metal5 left metal6 right metal6} …` instead of `add_rings -nets {VSS VDD} -layer {top {metal5 metal5} bottom {metal5 metal5} left {metal6 metal6} right {metal6 metal6}} …`
- Per Old behavior:

  ```
  add_rings -nets {VSS VDD} -width_bottom 5 -width_top 5 -width_right 5 -width_left 5 -layer_top Metal5 -layer_bottom Metal5 -layer_left Metal6 -layer_right Metal6 -offset_top 2 -offset_bottom 2 -offset_left 2 -offset_right 2 -spacing_top 1 -spacing_bottom 1 -spacing_left 1 -spacing_right 1
  ```

- Per New behavior:

  ```
  add_rings -nets {VSS VDD} -width 5 -layer {top Metal5 left Metal6 bottom Metal5 right metal6} -offset 2 -spacing 1
  ```

Related Topics

Power Planning and Routing chapter in the *Innovus User Guide*

add_rings Category Attributes

The root attributes in the `add_rings` category are:

- add_rings_avoid_short
- add_rings_break_core_ring_io_list
- add_rings_continue_on_no_selection
- add_rings_detailed_log
- add_rings_extend_block_ring_search_distance
- add_rings_extend_core_ring_search_distance
- add_rings_extend_merge_with_pre_wires
- add_rings_extend_over_row
- add_rings_extend_search_nets
- add_rings_extend_stripe_search_distance
- add_rings_gap_width_without_io
- add_rings_ignore_rows
- add_rings_max_via_size
- add_rings_orthogonal_only
- add_rings_skip_shared_inner_ring
- add_rings_skip_via_on_pin
- add_rings_skip_via_on_wire_shape
- add_rings_spacing_from_block
- add_rings_split_long_via
- add_rings_stacked_via_bottom_layer
- add_rings_stacked_via_top_layer
- add_rings_target
- add_rings_via_using_exact_crossover_size
- add_rings_wire_center_offset

To set the value of any root attribute, use the `set_db` command:

```
set_db attribute_name value
```

To get the current value of any root attribute, use the `get_db` command:

```
get_db attribute_name
```

To get the current value of all attributes in the `add_rings` category, use the following command:

```
get_db -category add_rings
```

add_rings_avoid_short

```
add_rings_avoid_short {true | false}
```

Default: `false`

Data_type: bool, read/write

Specifies whether to prevent a ring or a ring segment from being created if it would cause a short violation to occur. Specify `true` to prevent the creation of rings that would cause a short violation. Specify `false` to enable the creation of rings that cause a short violation.

add_rings_break_core_ring_io_list

```
add_rings_break_core_ring_io_list io_list
```

Default: `""`

Data_type: string, read/write

Breaks or excludes core rings for specific instances.

add_rings_continue_on_no_selection

```
add_rings_continue_on_no_selection {true | false}
```

Default: `false`

Data_type: bool, read/write

If the design has no power domain, or no fences are selected, `add_rings` ignores the option `-around power_domain` and generates core rings.

add_rings_detailed_log

```
add_rings_detailed_log {true | false}
```

Default: `false`

Data_type: bool, read/write

Setting this option will make `add_rings`, `addStripe` output an extensive log or a detailed log.

add_rings_extend_block_ring_search_distance

```
add_rings_extend_block_ring_search_distance distance
```

Default: `false`

Data_type: double, read/write

Extends the block ring search a specified distance beyond the initial block ring target to enable the connection of multiple block ring targets. For example, if VDD consists of three bits (segments) running parallel to one another, each having a width of two microns and a spacing between them of one micron, using this variable with a specified distance of eight microns will extend the block ring search a distance of eight microns to enable a connection to the multiple block ring targets. The *distance* is specified as the measured distance from the first to last bits. Units in microns.
Note: You must specify this variable before using the `add_rings` command. When using the command, you must include the following attributes:

```
-use_wire_group_bits 1
-use_wire_group 1
```

add_rings_extend_core_ring_search_distance

```
add_rings_extend_core_ring_search_distance distance
```

Default: `0.0`

Data_type: double, read/write

Specifies the wire group search distance for connecting all bits of the wire group stripes.

add_rings_extend_merge_with_pre_wires

```
add_rings_extend_merge_with_pre_wires {true | false}
```

Default: `true`

Data_type: bool, read/write

Automatically merges newly-created ring wires with samenet prerouted wires.

add_rings_extend_over_row

```
add_rings_extend_over_row {true | false}
```

Default: `false`

Data_type: bool, read/write

If `true` is specified, the cut row is not considered a blockage by the add_rings command, which generates the ring over the cut row. If `false` is specified, the cut row is considered a blockage and the ring is not generated over the row. Rings are created based on the attributes that you specify in the `add_rings` command.

add_rings_extend_search_nets

```
add_rings_extend_search_nets {* | {net_names}}
```

Default: `""`

Data_type: string, read/write

Specifies the net names to append to the wire group search distance properties.

add_rings_extend_stripe_search_distance

```
add_rings_extend_stripe_search_distance distance
```

Default: `0.0`

Data_type: double, read/write

Specifies the wire group search distance for connecting all bits of the wire group core rings.

add_rings_gap_width_without_io

```
add_rings_gap_width_without_io width
```

Default: `false`

Data_type: double, read/write

Specifies the maximum missing IO pads and the area width between IO pads.

add_rings_ignore_rows

```
add_rings_ignore_rows {true | false}
```

Default: `false`

Data_type: bool, read/write

Specifies whether the power planning software ignores all rows (cut rows and native rows) when rings are created. Specify `true` to create rings that ignore rows. Specify `false` to create rings that do not ignore rows.

add_rings_max_via_size

```
add_rings_max_via_size { shape width% height% target_penetration% }
```

Default: `""`

Data_type: string, read/write

The maximum size of a crossover via, specified as a percentage of a full-size via. You must specify three values for each shape that you specify. The shape values are:

- Pad
- Ring

- Stripe
- Blockring
- Blockpin
- Cover
- Noshape

The three values you must specify for each shape are:

- The maximum width for the crossover via, as a percentage of a full-size via
- The maximum height for the crossover via, as a percentage of a full-size via
- The target penetration, which specifies a smaller size for vias connecting to targets of the specified shape

add_rings_orthogonal_only

```
add_rings_orthogonal_only {true | false}
```

Default: If you do not specify this attribute or if you specify it with a value 1, the software only creates vias for orthogonal wires and pin intersections.

Data_type: bool, read/write

Creates vias *only* for orthogonal wires and pin intersections.

add_rings_skip_shared_inner_ring

```
add_rings_skip_shared_inner_ring {horizontal | vertical | none}
```

Default: `none`

Data_type: enum {horizontal vertical none}, read/write

Removes the horizontal or vertical portion of rings (shared or inside edges) when used in conjunction with the add_rings -block_rings -around shared_cluster command.

add_rings_skip_via_on_pin

```
add_rings_skip_via_on_pin {[Pad] [Block] [Cover] [Standardcell] [ Physicalpin]}
```

Default: If you do not specify this attribute, vias are generated for block pins, pad pins, and cover macro pins, but are not generated for standard cell pins.

Data_type: enum* {pad block cover standardcell physicalpin}, read/write

Prevents vias from being generated on the specified types of pins.

If you specify `-skip_via_on_pin {}`, the power planning software makes connections to all five categories of pins.

add_rings_skip_via_on_wire_shape

```
add_rings_skip_via_on_wire_shape {blockring | stripe | followpin | corewire | blockwire | iowire | padring | ring | fillwire | noshape}
```

Default: If you do not specify this attribute, vias can be generated on all wire shapes *except* no-shape wires.

Data_type: enum* {blockring stripe followpin corewire blockwire iowire padring ring fillwire noshape}, read/write

Prevents vias from being generated on the specified wire shapes. If you specify `-skip_via_on_wire_shape {}`, the power planning software makes connections to all five categories of wire shapes.

add_rings_spacing_from_block

```
add_rings_spacing_from_block value
```

Default: 0

Data_type: double, read/write

To be used together with `-break_core_ring_io_list` *`io list`* to specify the spacing between broken wire end and the io cell.

add_rings_split_long_via

`add_rings_split_long_via { threshold_value step_value offset_value height_value }`

Default: If you do not specify this attribute, the software does not split vias.

Note: If the offset_value is given as negative, the vias will be placed symmetrically on both ends. The default value of `offset_value` is `-1` .

Data_type: string, read/write

Splits vias longer than the specified length (`threshold_value` , in micrometers) into smaller vias with specified center-to-center spacing (`step_value` , in micrometers), left/bottom end offset (`offset_value` , in micrometers) and vertical/horizontal height (`height_value` , in micrometers) values . This is independent of the option `-split_via`.

add_rings_stacked_via_bottom_layer

`add_rings_stacked_via_bottom_layer` *`layername`*

Default: The lowest metal layer in the design.

Data_type: string, read/write

Specifies the lowest layer in which vias can be stacked.

add_rings_stacked_via_top_layer

`add_rings_stacked_via_top_layer` *`layername`*

Default: The highest metal layer in the design.

Data_type: string, read/write

Specifies the highest layer in which vias can be stacked.

add_rings_target

`add_rings_target {core_ring | default | first_ring | pad_ring | stripe}`

Default: `default`

Data_type: enum {core_ring default first_ring pad_ring stripe}, read/write

The add_rings_target attribute consists of the following attributes:

- core_ring: Ring wires can consider core rings as targets, even when core rings are inside the I/O pad area. Only rings are considered targets.
- pad_ring: Ring wires can be extended to pad rings.
- first_ring: Ring wires consider either the first core ring or the pad ring as a target.
- stripe: Both rings and stripes are considered targets.

Note: You can specify that rings are extended to both pad rings and to stripes. To do this you must specify the ring_target attribute twice, as show in the following example:

```
-ring_target stripe
-ring_target pad_ring
```

add_rings_via_using_exact_crossover_size

`add_rings_via_using_exact_crossover_size {true | false}`

Default: `false`

Data_type: bool, read/write

Specifies whether to generate partial vias of the exact crossover size. A value of `false` allows partial vias to be generated. A value of `0` prevents partial vias, and the software always generates full-size vias.

Note: This option does not apply when both of the objects at the crossover are rings or stripes, or at the crossover of a ring and stripe. In these cases, the software always generates a full-size via array.

add_rings_wire_center_offset

```
add_rings_wire_center_offset {true | false}
```

Default: `false`

Data_type: bool, read/write

Used for calculating the offset. When set to `true`, `add_rings` will use wire center. When set to `false`, `add_rings` will use wire boundary.

add_split_power_vias

```
add_split_power_vias
[-help]
[-delete | -wires_only]
[-edge_length {layer1 value1 layer2 value2 ...}]
[-force_count]
[-local_checker_board]
[-master cell_name]
[-max_via_size {x y}]
[-max_wire_length {layer1 value1 layer2 value2 ...}]
[-max_wire_spacing {layer1 value1 layer2 value2 ...}]
[-merge_count {x y}]
[-merge_distance {x y}]
[-min_enclosure]
[-min_via_size {x y}]
[-min_wire_length {layer1 value1 layer2 value2 ...}]
[-min_wire_spacing {layer1 value1 layer2 value2 ...}]
[-pin_layer {layer1 layer2 ...}]
[-preroute_align {none samelayer alllayer}]
[-snap_wire_center_to_grid {layer1 {none|grid|half_grid|either} layer2 {none|grid|half_grid|either}} ...]
[-stacked_checker_board]
[-user_class userDefinedAttribute]
[-wire_count {layer1 value1 layer2 value2 ...}]
[-wire_offset {layer1 value1 layer2 value2 ...}]
[-wire_width {layer1 value1 layer2 value2 ...}]
```

Adds or deletes Jenga vias and does not modify vias. It passes on all Jenga parameters to separate options.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `add_split_power_vias` parameter. For a detailed description of the command and all its parameters, use the `man` command: `man add_split_power_vias`
`-delete`	Deletes the via for selected wires/objects. This is a boolean value and is optional. *Data_type*: bool, optional

`-edge_length {layer1 value1 layer2 value2 ...}`	Sets extension length of the wires for each layer in um. The value is a string and is optional. *Data_type*: string, optional
`-force_count`	Forces the tool to generate the given wirecount even if the wires extend beyond the overlap area. *Data_type*: bool, optional
`-local_checker_board`	Allows local checkerboard cut generation instead of matching the global checkerboard pattern. *Data_type*: bool, optional
`-master cell_name`	Passes the selected wires through the specified cells. *Data_type*: string, optional
`-max_via_size {x y}`	Displays the maximum via size. *Data_type*: point, optional
`-max_wire_length {layer1 value1 layer2 value2 ...}`	Sets the maximum length of the wires for each layer in um. *Data_type*: string, optional
`-max_wire_spacing {layer1 value1 layer2 value2 ...}`	Sets the maximum wire spacing of wires for each layer in um. *Data_type*: string, optional
`-merge_count {x y}`	Merges the neighboring wire count. *Data_type*: int, optional
`-merge_distance {x y}`	Merges the distance in x and y direction. *Data_type*: point, optional
`-min_enclosure`	Creates the shortest wires that meet DRC rules rather than the default wire length of the entire overlap area. *Data_type*: bool, optional
`-min_via_size {x y}`	Sets the minimum via size. *Data_type*: point, optional
`-min_wire_spacing {layer1 value1 layer2 value2 ...}`	Sets the minimum wire spacing of wires for each layer in um. *Data_type*: string, optional
`-min_wire_length {layer1 value1 layer2 value2 ...}`	Sets the minimum length of the wires for each layer in um. *Data_type*: string, optional
`-wire_count {layer1 value1 layer2 value2 ...}`	Sets the count of wires for each layer in integer. *Data_type*: string, optional
`-wire_offset {layer1 value1 layer2 value2 ...}`	Sets the wire offset away from the intersection of selected wires. *Data_type*: string, optional
`-wire_width {layer1 value1 layer2 value2 ...}`	Sets the width of the wires for each layer in um. *Data_type*: string, optional

`-pin_layer {layer1 layer2 ...}`	Lists pin layers in a macros used to create via with crossover wire. *Data_type*: string, optional
`-preroute_align {none | samelayer | alllayer}`	Defines the alignment of via wires with preexisting wires on the same net. *Data_type*: enum, optional
`-snap_wire_center_to_grid {layer1 {none|grid|half_grid|either} layer2 {none|grid|half_grid|either}} ...`	Specifies the wire center alignment style for middle layer jenga wires. If you do not specify this parameter, stripes are not snapped to the routing grid. Specify one of the following options: • `None`: Does no snapping of the wires. • `Grid`: Snaps the center of the wire to the routing grid of the same layer as the wire. • `Half_Grid`: Snaps the center of the wire to the half-routing grid. The half-routing grid is a virtual grid with the tracks at the center of every two adjacent tracks of the routing grid. • `Either`: Snaps the center of the wire to the closer of the routing grid or half-grid track. *Data_type*: string, optional
`-stacked_checker_board`	Creates stacked checker board vias between intermediate layers. *Data_type*: bool, optional
`-user_class userDefinedAttribute`	Adds the user-defined attribute to the to-be-built split vias and wires. *Data_type*: string, optional
`-wires_only`	Generates only those wires, which are without violations. By default, the command generates both wires and vias. Prior to implementing this option, if any via was not dropped because of violation, the tool was not generating any wires. *Data_type*: bool, optional

Examples

Note: The command will add split power via after the user selects the wires which need to add the split power via (else, no split power via will be added).

- The command will add split power via and merge wires which are within 1.2 for both x and y direction
  ```
  add_split_power_vias -merge_distance 1.2
  ```
- The command will add split power via and merge wires which are within 1.2 for x direction and 1.5 for y direction.
  ```
  add_split_power_vias -merge_distance { 1.2 1.5 }
  ```
- The command will add split power via and merge wires of count 2 in both x and y direction.
  ```
  add_split_power_vias -merge_count 2
  ```
- The command will add split power via and merge wires of count 2 in x direction and count of 4 in y direction.

```
add_split_power_vias -merge_count { 2 4}
```

- The command will add split power via and create wire lengths to a max of 1.3 in both x and y direction for all intermediate layers within a crossover b/w wire and a wire or b/w wire and a pin.

```
add_split_power_vias -max_wire_length 1.3
```

Related Topics

Power Planning and Routing chapter in the *Innovus User Guide.*

add_stripes

```
add_stripes
[-help]
[-all_blocks {0 | 1}]
[-area {x1 y1 x2 y2 ...}]
[-area_blockage {x1 y1 x2 y2}... {x1 y1 x2 y2 x3 y3 x4 y4 ...} ...]
[-between_bumps {0 | 1}]
[-block_ring_bottom_layer_limit layer]
[-block_ring_top_layer_limit layer]
[-create_pins {0 | 1}]
[-direction {horizontal | vertical}]
[-extend_to {design_boundary | first_padring | last_padring | all_domains}]
[-insts instance_name]
[-layer layer]
[-master master_cell]
[-max_same_layer_jog_length real_value]
[-merge_stripes_value {auto|value}]
[-narrow_channel {0 | 1}]
[-nets list_of_nets]
[-number_of_sets integer_value]
[-over_bumps {0 | 1}]
[-over_physical_pins {0 | 1}]
[-over_pins {0 | 1}]
[-over_power_domain {0 | 1}]
[-pad_core_ring_bottom_layer_limit layer]
[-pad_core_ring_top_layer_limit layer] [-pin_layer layer]
[-pin_offset real_value]
[-pin_width { min_value max_value }]
[-power_domains domain_name]
[-report_cut_stripe  filename ]
[-same_layer_target_only {0 | 1}]
[-set_to_set_distance real_value]
[-snap_wire_center_to_grid {none | grid | mask1_grid | mask2_grid | half_grid | either} [-
allow_snapping_override_custom_spacing {0 | 1}]
[-spacing name_or_value]
[-stapling {length | auto {offset layer1_pitch:n | reference_layer1 [reference_layer2]}]
[-start real_value]
[-start_from {left | right | bottom | top}]
[-start_offset real_value]
[-stop real_value]
[-stop_offset real_value] [-switch_layer_over_obs {0 | 1}]
[-user_class subclass_string] [-use_interleaving_wire_group {0 | 1}]
[-use_wire_group {-1 0 1}]
[-use_wire_group_bits integer_value]
[-via_rows integer]
[-via_columns integer]
[-width name_or_value]
```

Creates power stripes within the specified area. If the router encounters an obstruction, the stripe connects to the last stripe on the same net. Otherwise, the stripes stop at the core row boundary. Use this command after creating an initial floorplan.

Adding Stripes in Multi-Threading Mode

You can generate stripes in multi-threading mode by using the following command before `add_stripes`:

```
set_multi_cpu_usage
```

When the specified CPU number is bigger than the estimated CPU number, `add_stripes` uses the estimated CPU number for multi-threading. Otherwise, it uses the specified CPU number.

Estimated CPU number = (free memory – 512 MB)/(added memory*2)

If any of the following options are specified, `add_stripes` will internally disable the multi-threading mode:

- `{-over_bumps {0 | 1} | -between_bumps {0 | 1}} | -over_pins {0 | 1} -pin_layer layer {-master master_cells | -all_blocks}}`
- UDM global variable `pwStripeRouteOverRowsOnly` is equal to 1

`add_stripes` supports the new `set_multi_cpu_usage` command use model.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `add_stripes` parameter. For a detailed description of the command and all its parameters, use the `man` command: `man add_stripes`
`-all_blocks {0 | 1}`	Specifies that stripes should be generated over all blocks. *Default:* `0` *Data_type*: bool, optional
`-allow_snapping_override_custom_spacing {0 | 1}`	When set to `1`, it allows snapping of stripes overriding the specified spacing to the closest track. If the closest track has conflicts, the stripe is removed. When set to `0`, it snaps the stripes in the first set to the inward direction, and the other stripes to the closest track. This parameter keeps the actual spacing greater than or equal to the specified spacing. *Default:* `0` *Data_type*: bool, optional
`-area {x1 y1 x2 y2...xn yn}`	Specifies the coordinates of the design area in which the stripes must be created. The `-x_left_offset` and `-x_right_offset` values are measured from this boundary area for vertical stripes, and the `-y_top_offset` and `-y_bottom_offset` values are measured from this boundary area for horizontal stripes. **Note:** • If a core ring exists in the design, the stripes automatically start and stop at the core ring if you do not use the `-area` parameter. You can also create stripes in an area defined by a polygon (rectilinear). For example: `add_stripes -area {100 100 100 200 150 200 150 150 200 150 200 100}` *Default:* If you do not specify this parameter, stripes are created for the entire design area. • `-area` and `-extend_to design_boundary` are mutually exclusive. The following error is issued on them being used together: *You cannot specify stripes over design boundary and set an area constraint. Instead, specify area to cover design boundary.* • `-area` and `-extend_to first_padring/ last_padring` are mutually exclusive. The following error is issued on them being used together: *You cannot specify stripes over padring and set an area constraint. Instead, specify area to cover padring.* • When this parameter is used with `add_stripes_domain_offset_from_core true`, the `-start_offset` value for specified area is measured from core on using `add_stripes -area`. *Data_type*: string, optional

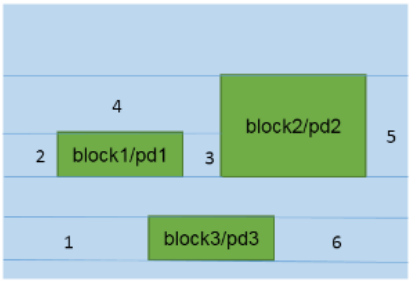
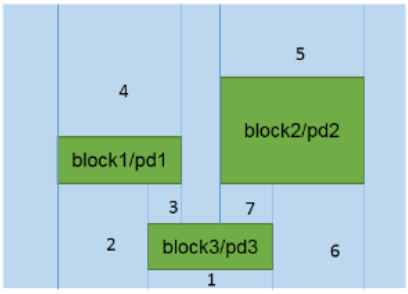
`-area_blockage {{x1 y1 x2 y2...}...{xn yn}}`	Specifies the blockage coordinates to restrict the addition of any stripes to this region.
	This parameter supports rectangular and rectilinear area_blockages. Their spacing is 0 during stripe generation.
	Examples
	• Rectangular area_blockage:
	`-area_blockage {100 100 200 200}`
	• Rectilinear area_blockage:
	`-area_blockage {100 100 100 200 150 200 150 150 200 150 200 100}`
	You can add multiple blockages at the same time.
	Rectangular areas or rectilinear area or their mixture can be specified. For multiple areas, each of them must be inside brace.
	Examples
	• multiple rectangles
	`-area_blockage {{x1 y1 x2 y2} {a1 b1 a2 b2} ...}`
	• multiple rectilinear area
	`-area_blockage {{x1 y1 x2 y2 ... } {a1 b1 a2 b2 a3 b3 ...} ...}`
	• mixture of them
	`-area_blockage {{x1 y1 x2 y2} {a1 b1 a2 b2} ... {x1 y1 x2 y2 x3 y3 ...} {a1 b1 a2 b2 a3 b3 ...} ...}`
	The specified areas that are not included inside brace will be ignored. For example
	`-area_blockage {{x1 y1 x2 y2} a1 b1 a2 b2 a3 b3 ... {c1 d1 c2 d2 c3 d3 ...}}`
	`{x1 y1 x2 y2}` and `{c1 d1 c2 d2 c3 d3 ...}` will be used, and `a1 b1 a2 b2 a3 b3 ...` are ignored
	Data_type: string, optional

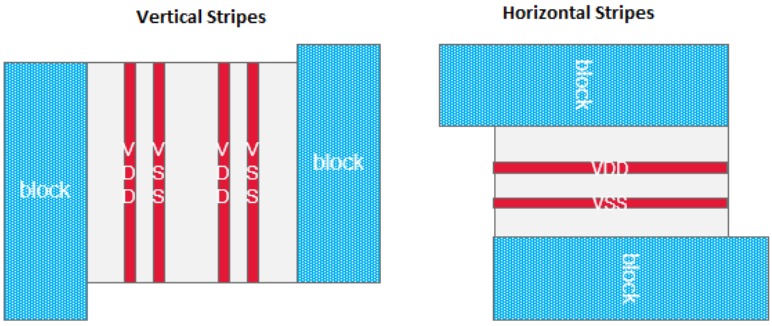
`-between_bumps {0 | 1}`	Specifies that stripes should be generated between bumps. *Default*: `0` *Data_type*: bool, optional
`-block_ring_bottom_layer_limit layer_name`	Specifies the lowest layer that stripes can switch to when encountering a block ring. Stripes that encounter a block ring can only switch to this layer or higher. *Data_type*: string, optional
`-block_ring_top_layer_limit layer_name`	Specifies the highest layer that stripes can switch to when encountering a block ring. Stripes that encounter a block ring can only switch to this layer or lower. *Data_type*: string, optional
`-create_pins {0 | 1}`	If set to `1`, IO pins are created for each stripe. Any overlap with existing IO pins at the design boundary is flagged as a warning after the stripes are created. If set to `0`, pins are not created. By default, `add_stripes` would create pins for any stripe touching the design boundary. When `-extend_to design_boundary` is used the option `-create_pins` is 1 by default. **Note:** The switch `-create_pins` does not work if you also specify the `-area` parameter. It is difficult for the user to understand the original. *Data_type*: bool, optional

`-direction {horizontal | vertical}`	Sets the stripe direction. *Default:* If you do not specify this parameter, vertical stripes are created. *Data_type*: enum, optional

`-extend_to`	The boundary for the stripes. Specify one of the following values:
	`all_domains`
	Stripes are created over all power domains for each power and ground net within each power domain. Stripes start and stop at each power domain boundary or ring around the power domain, and ignore the values in the `-nets` parameter.
	`first_padring`
	Stripes are created within the area enclosed by the pad ring closest to the core area.
	Stripes are created within the area enclosed by the pad ring closest to the design boundary.
	`design_boundary`
	Stripes are created within the design boundary
	Data_type: enum, optional

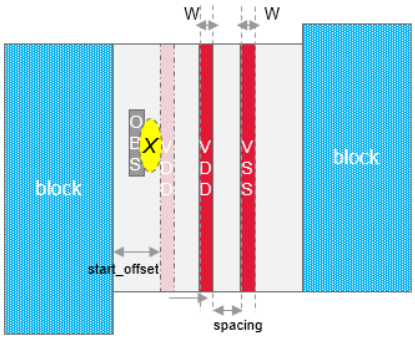
`-insts` *instance_name*	Specifies the instance name for generating power stripes. *Data_type*: string, optional
`-layer` *layer*	Specifies the name of the layer in which to create stripes. Stripes can be created on only one layer at a time. The following types of formulas are supported: 1. Lefname 2. Metal layer number **Example:** If Metal2 is named by 'MET2' in lef file, then you can use 1. MET2 2. 2 to identify metal2. *Data_type*: string, required
`-master` *master_cells*	Specifies the instances of the standard cell (or block) masters over which stripes should be generated. The option supports regular expression: `?` matches any single characters and `*` matches any sequence of characters. *Data_type*: string, optional
`-max_same_layer_jog_length` *value*	The maximum length, in micrometers, that a stripe can jog on the same layer before switching to an adjacent layer. For example, if you specify a value of 2 micrometers, and an obstruction causes a stripe on layer METAL4 to jog for 3 micrometers, the jog will occur on layer METAL3 or METAL5. However, if an obstruction causes the stripe to jog for only 1 micrometer, the jog occurs on layer METAL4. *Default:* `2` (If you do not specify this parameter, the stripe automatically switches to an adjacent layer if the jog length is two times the width of the stripe.) *Data_type*: float, optional

`-merge_stripes_value` *value*	Merges a stripe with a nearby block ring if the threshold spacing between the stripe and ring is smaller than the *value* (in micrometers) specified with this parameter. The stripe jogs to connect to the nearby ring only on the stripe layer. **Note:** To prevent stripes from merging with block rings, set the value of this parameter to –1. *Default:* 10 (If you do not specify this parameter, stripes do not jog to merge with rings that are more than 10 µm away.) To merge the stripe on all layers, set the following variable of the `set_db` command: `-merge_with_all_layers 1` *Data_type*: string, optional
`-narrow_channel {0 | 1}`	Generates stripes over narrow channel area. Narrow channel usually exists between two blocks/power domains, or between core boundary and blocks/power domains. If there are part of rows without a certain net and layer of power mesh crossing over, the area of neighboring part of rows is defined as a narrow channel for that net and layer. When adding overall stripes, some narrow channels might be missed because of start offset and set to set distance. In this circumstance, adding stripes in narrow channel can help to supply power to stdcells in that channel. The recommended flow is to add overall stripes first and then to add stripes in narrow channel, if needed. *Default*: `0` **Examples of narrow channels:** Narrow potential channels for vertical stripes  Narrow potential channels for horizontal stripes  **Note:** This parameter can be used with the `-power_domains` and `-over_power_domain` parameters. The `-narrow_channel` parameter generates stripes with uniform channel width in both the vertical and horizontal directions, if the `-number_of_sets` parameter is set to an integer value greater than `1`. The added wires are created at the channel evenly, as shown below:



If the `-start_offset` parameter is used for specifying the starting offset value before the above setting of the `-narrow_channel` parameter, and the specified offset location has a DRC, the added wire shifts to the left or up to the closed DRC clean location, as shown below:

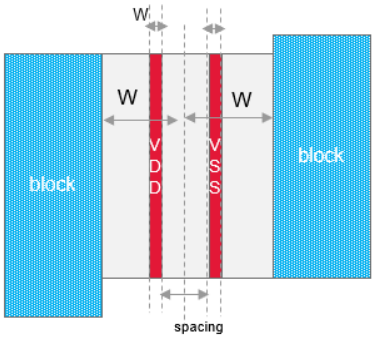
```
add_stripes -width $W -spacing $spacing -number_of_sets N -narrow_channel 1 -nets
{$net1 $net2 ...} ...
```



If the `-start_offset` parameter is not used before the above setting of the `-narrow_channel` parameter, the wires are added to the center of the channel. With this setting:

- When the `-number_of_sets` parameter is set to `1`, this center has the same distance from the channel boundaries, as shown below:

  ```
  add_stripes -width $W -spacing $spacing -number_of_sets 1 -narrow_channel 1 -nets
  {$net1 $net2 ...} ...
  ```

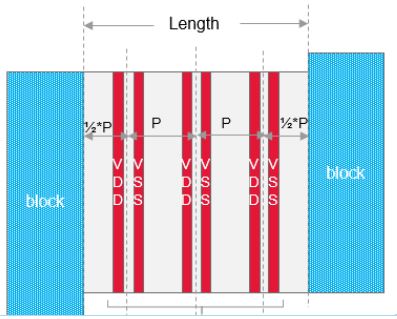
  

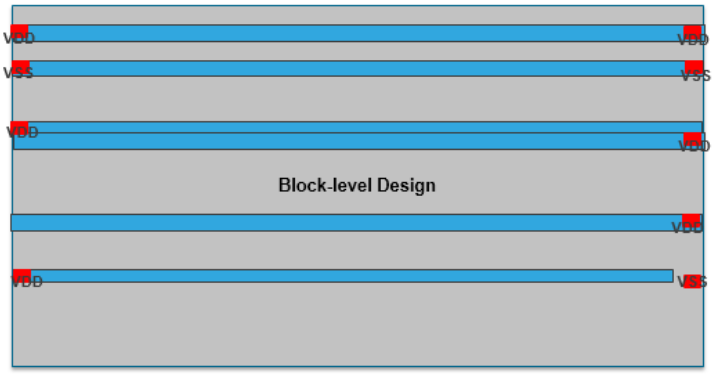
- When the `-number_of_sets` parameter is set to an integer value (`N`) greater than `1`, `add_stripes` calculates the pitch to evenly distribute the PG in the channel.

  ```
  Pitch: P = Length / N
  ```

  Additionally, the two PG stripes set at the edge of the channel are at the same distance from the middle of the PG set and the channel boundary is at `P/2`, as shown below:

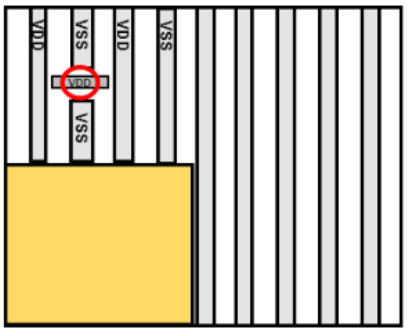
  ```
  add_stripes -width $W -spacing $spacing -number_of_sets $N -narrow_channel 1 -
  nets {$net1 $net2 ...} ...
  ```

	
`-nets {list_of_nets}`	Specifies the names of the nets for which stripes are to be created. The number of net names determines the number of stripes within each set of stripes. The first net in the name list is created first and corresponds to the left or bottom stripe of the set. Separate multiple net names with a space and enclose the list in curly braces, for example, `-nets {net1 net2 net3 net3}`. Braces are not required for one name. *Data_type*: string, required
`-number_of_sets integer`	Specifies the number of stripe sets to be created. You must specify either this parameter or the `-set_to_set_distance` parameter. *Default:* If you do not specify this parameter, the number of sets is derived from the `-set_to_set_distance` parameter. *Data_type*: int, optional
`-over_bumps {0 | 1}`	Specifies that stripes should be generated over bumps. *Default*: `0` *Data_type*: bool, optional

`-over_physical_pins {0 | 1}`	Adds stripes over the physical pins to detect the physical PG terminals and add stripes to continue them. Following is the use model of this option:
	`add_stripes -over_physical_pins 1 -nets {net_names} -width {pin_width|width_value} -direction {vertical|horizontal} [-pin_layer Mx] [-pin_offset offset_value]`
	While using this option:
	◦ The tool generates one stripe on one physical pin separately, except for the following special cases:
	▪ When two same net pins (left and right) are totally aligned. In this case, one stripe between two pins is generated.
	▪ When two different net pins are partially or totally aligned. In this case, one stripe on left/bottom pin is generated and break at right/up pin with DRC clean distance.
	◦ `-width pin_width` makes stripe width same as that of related physical P/G pin.
	◦ When `offset_value` in `-pin_offset offset_value` is greater than zero, stripes shift up for left/right pins and right for up/down pins.
	◦ If both `-over_physical_pins` and `-over_pin` are specified, `-over_physical_pins` is ignored.
	In the following example, stripes are added over the physical pins in the horizontal direction for the `vdd` and vss nets:
	`add_stripes -over_physical_pins 1 -nets {VDD VSS} -width 10 -direction horizontal -pin_layer metal11`
	Following is the block-level design of the output:
	 
	
	Data_type: bool, optional

`-over_pins {0 | 1}`	Specifies that stripes should be generated over power/ground pins in a standard cell or block. *Default*: `0` **Note:** With this option, user could define value `pin_width` in option `-width`, then tool will generate stripe according to pin width automatically. *Data_type*: bool, optional

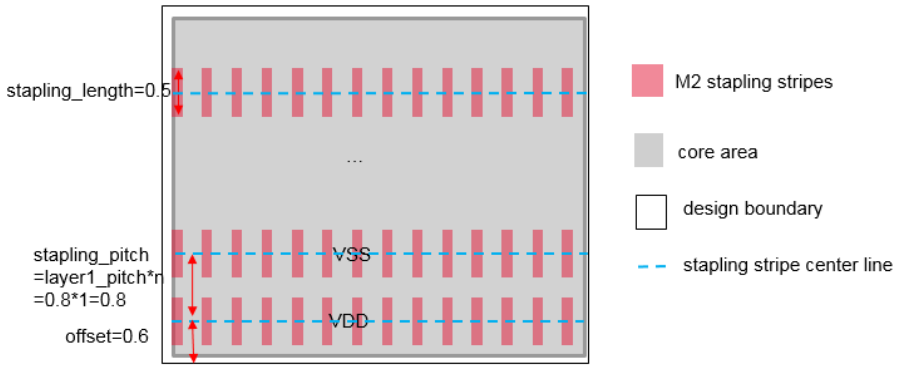
`-over_power_domain {0 | 1}`	If set to `1`, adds stripes only within power domain boundaries or the fence around a soft block. For power domains, these stripes start and stop at the power ring that surrounds the selected power domain boundary. For fences, these stripes start and stop at the fence. If both a power domain and a fence are selected, the power domain takes precedence. If you do not specify this parameter or if you specify it with a value of `0`, stripes are prevented from extending to the power domain boundary. **Note:** • When this parameter is specified, you can use `-start_from {bottom | top | left | right}` and `-start_offset` parameters to set boundaries, but you cannot use the `-start` and `-stop` parameters to specify an absolute location. • In addition, the software issues a warning message if the specified power and ground nets do not match the power domain's power or ground nets. • When `add_stripes -over_power_domain 1` is used with `add_stripes_domain_offset_from_core true`, the `-start_offset` value for the selected power domain/block is measured from core on using `add_stripes -over_power_domain 1`. *Default:* `0` *Data_type*: bool, optional
`-pad_core_ring_bottom_layer_limit` *layer_name*	Specifies the lowest layer that stripes can switch to when encountering a pad or core ring. Stripes that encounter a pad or core ring can only switch to this layer or higher. *Data_type*: string, optional
`-pad_core_ring_top_layer_limit` *layer_name*	Specifies the highest layer that stripes can switch to when encountering a pad or core ring. Stripes that encounter a pad or core ring can only switch to this layer or lower. *Data_type*: string, optional
`-pin_layer` *layer*	Specifies the layer on which the stripe is generated. **Note:** The layer you specify with this parameter is the same as the layer you specify with the `-layer` parameter. *Default*: `TOP` *Data_type*: string, optional
`-pin_offset`	Offsets the stripe a specified distance from the pin edge to make well tap cell connections. The offset is from the left edge of the stripe to the left edge of the same net pin. This can be a negative number and is in user units. *Data_type*: float, optional
`-pin_width {`*min_value max_value*`}`	Specifies the range of minimum and maximum pin width for `-over_pins` or `-over_physical_pins`. *Data_type*: string, optional
`-power_domains` *domain_name*	Specifies the power domain name for generating power stripes. *Data_type*: string, optional

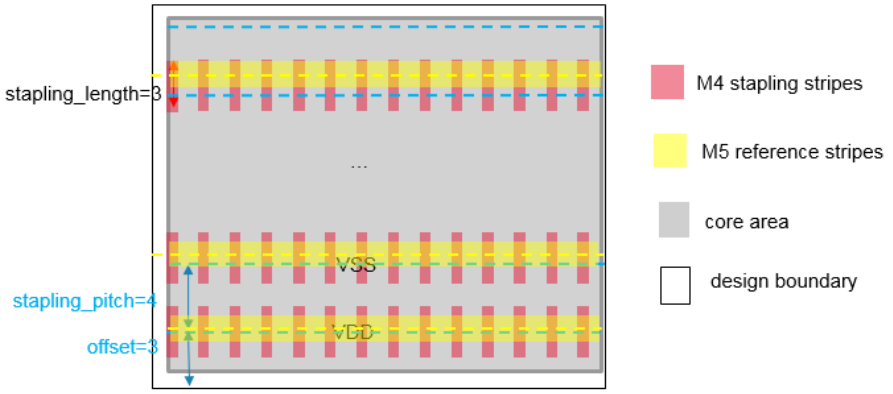
`-report_cut_stripe` *filename*	Saves the information related to a cut stripe in a DRC file. This helps in identifying the power stripe that has been removed/cut from the design to avoid any DRC violation following user settings.
	You can use the following command to load the generated DRC file into the design and view the cut stripe mark in the GUI:
	`read_drc_markers *.drc`
	The following example shows how the information related to a cut stripe is stored in the a.drc file:
	`add_stripes -report_cut_stripe a.drc`
	The following figure depicts that the place highlighted in red circle will be marked, because here the VSS stripe is cut to avoid short violation to VDD segment:
	 
	Data_type: string, optional

`-same_layer_target_only`	Specifies the connection to overlapping targets on the same layer only. *Data_type*: bool, optional
`-set_to_set_distance` *real*	Specifies the distance (pitch) from the reference stripe of one set to the reference stripe of the next set. The value of this parameter must be a positive number specified in micrometers. You must specify either this parameter or the `-number_of_sets` parameter. *Default:* If you do not specify this parameter, the distance between sets is derived from the `-number_of_sets` parameter. *Data_type*: float, optional
`-snap_wire_center_to_grid {none | grid | mask1_grid | mask2_grid | half_grid | either}`	Controls the snapping of the center of the stripes. If you do not specify this parameter, stripes are not snapped to the routing grid. Specify one of the following options: • `none`: Does no snapping of the wires. • `grid`: Snaps the center of the wire to the routing grid of the same layer as the wire. • `mask1_gird`: Snaps the center of the wire to the routing track of Mask1 . • `mask2_grid`: Snaps the center of the wire to the routing track of Mask2 . • `half_grid`: Snaps the center of the wire to the half-routing grid. The half-routing grid is a virtual grid with the tracks at the center of every two adjacent tracks of the routing grid. • `either`: Snaps the center of the wire to the closer of the routing grid or half-grid track. *Data_type*: enum, optional
`-spacing` *spacing*	Specifies the edge-to-edge spacing between stripes in each set. You can specify different spacing between each pair of stripes. For *n* number of stripes, you can specify *n* –1 spacing values. If fewer values are specified, the last-specified spacing is used for the remaining unmatched stripe pairs. *Data_type*: string, optional

`-stapling {length | auto {offset layer1_pitch:n | reference_layer1 [reference_layer2]}`

Turns on the stapling stripe generation feature and controls the stapling pattern. In the syntax:

- `length`: Specifies the length of stapling stripes.
- `auto`: When specified, stapling length is calculated by the tool automatically, with the minimum length that can be DRC clean.
- `offset`: Specifies the distance from design boundary to the center of the first stapling stripe by default. Refer to the following example:
  Example 1
  ```
  add_stripes -stapling {0.5 0.6 0.8:1} -nets {VDD VSS} -layer metal2 -width 0.1 -direction vertical -set_to_set_distance 0.5
  ```
  

- `pitch:n`: Specifies the center to center distance between two stapling stripes in end direction (refer to **Example 1** above). It is defined by `n` times of the track pitch on `layer1`, where stripes are needed to be connected by stapling stripes.
- `reference_layer1`: When the stripes on `layer1` to be connected already exist, you can directly specify the reference layer 1 name. It specifies that the stapling stripe center is on the center line of the reference stripes (offset value is the center offset of first bottom/left stripe; the `pitch:n` is the set-to-set distance of stripes on `layer1`).
- `reference_layer2`: When the stripes on `layer2` to be connected already exist, you can specify the reference layer 2 name, so that the stapling stripes are generated referring to both `layer1` and `layer2` and connect them. When `reference_layer2` is specified, the real offset and stapling pitch will be the average value from `layer1` and `layer2` settings. It aims to find the center of two layers of wires to be connected by stapling stripes. Refer to the example below:
  Example 2
  ```
  set metal3_track_offset 2
  set metal3_pitch 0.8
  add_stripes -stapling '3 $metal3_track_offset $metal3_pitch:5 metal5' -nets {VDD VSS} -layer metal4 -width 0.5 -direction vertical -set_to_set_distance 2
  ```
  

You can use `-stapling` in the following ways:

	• `-stapling {length|auto offset pitch:n}` • `-stapling {length|auto reference_layer}` • `-stapling {length|auto offset pitch:n reference_layer2}` • `-stapling {length|auto reference_layer1 reference_layer2}` *Data_type*: string, optional
`-start` *value*	The start coordinate for the region for adding the stripe. *Data_type*: float, optional
`-start_from {bottom | top | left | right}`	For horizontal stripes: • `bottom` indicates that stripes should be generated from bottom to top, taking the bottom offset into account. • `top` indicates that stripes should be generated from top to bottom, taking the top offset into account. *Default*: `bottom` For vertical stripes: • `left` indicates that stripes should be generated from left to right, taking the left offset into account. • `right` indicates that stripes should be generated from right to left, taking the right offset into account. *Default*: `left` **Note:** Offset values you set with this parameter are considered "hard": The power planning software adheres strictly to the values. However, the opposite values are considered "soft": Such values could be greater than the hard values, but will never be less than the hard values. For example, if you set a `right` offset value of 4 µm, the `left` value will be at least 4 µm, but could be greater. *Data_type*: enum, optional
`-start_offset` *value*	Specifies the starting offset value. This is an optional field. **Note:** The parameters `-start` and `-start_offset` are exclusive, and cannot be used together. *Data_type*: float, optional
`-stop`	The stop coordinate for the region for adding the stripe. *Data_type*: float, optional
`-stop_offset` *value*	Specifies the exact stop coordinates that contain the set of stripes to be created in the specified directions. *Default:* If you do not specify this parameter, stripes can be created anywhere in the design. *Data_type*: float, optional
`-switch_layer_over_obs {0 | 1}`	Controls whether a stripe switches layers, is routed over blocks, and then continues on the original layer. Using this parameter eliminates the need for a power routing mesh layer, and avoids maximum via stack rule violations. *Default*: 0 *Data_type*: bool, optional
`-user_class` *subclass_string*	Displays the user defined attribute attached to each created stripe segment. *Data_type*: string, optional

`-use_interleaving_wire_group {0 | 1}`	If set to 1, connects multiple wires from the same net together, even if separated by wires from a different net. You can only specify this parameter if you also specify `-use_wire_group 1`. If you do not specify this parameter or if you specify it with a value of `0`, interleaving wire groups are not used. *Default:* `0` *Data_type*: bool, optional
`-use_wire_group {-1 | 0 | 1}`	If set to `1`, connects wires from the same net together. If you do not specify this parameter or if you specify it with a value of `0`, wire groups are not used. If set to `-1` and multiple same net are specified in -nets, the wire group property will not be updated or created. **Note:** For stripes to connect to rings using wire groups, you must specify the `-use_wire_group` parameter in both the `add_rings` and `add_stripes` commands. *Default:* `0` *Data_type*: enum, optional
`-use_wire_group_bits` *integer*	Specifies how many nets are to be included in the wire group. You can only specify this parameter if you also specify `-use_wire_group 1`. *Default:* The same number of nets as specified in the `-nets` parameter. *Data_type*: int, optional
`-via_rows` *integer* `-via_columns` *integer*	Specify the number of cuts to make in the power vias (for both stack and non-stack). *Default*: `0` *Data_type*: int, optional
`-width` *width*	Specifies the width of the stripes. You can specify a different width for each net within a set. If the number of widths specified does not match the number of net names, the last-specified width is used for the rest of unmatched nets. *Data_type*: string, required

Example

The following command adds stripes in the vertical direction for the `vdd` and `gnd` nets.

```
add_stripes -direction vertical -nets {vdd gnd} -width 10 -spacing 1 -layer METAL6 -start_offset 50 -set_to_set_distance 50
```

Related Topics

Power Planning and Routing chapter in the *Innovus User Guide.*

add_stripes Category Attributes

The root attributes in the `add_stripes` category are:

- add_stripes_allow_jog
- add_stripes_allow_non_preferred_dir
- add_stripes_area
- add_stripes_blocks_without_same_net
- add_stripes_break_at
- add_stripes_continue_on_no_selection
- add_stripes_detailed_log
- add_stripes_domain_offset_from_core
- add_stripes_extend_to_closest_target
- add_stripes_extend_to_first_ring
- add_stripes_ignore_block_check
- add_stripes_ignore_block_ring_during_break
- add_stripes_ignore_drc
- add_stripes_ignore_non_default_domains
- add_stripes_in_cell_allow_shift
- add_stripes_in_cell_only
- add_stripes_keep_pitch_after_snap
- add_stripes_mask_color_balance
- add_stripes_max_extension_distance
- add_stripes_max_via_size
- add_stripes_merge_with_all_layers
- add_stripes_mesh_via
- add_stripes_offset_from_core
- add_stripes_opt_stripe_for_routing_track
- add_stripes_orthogonal_offset
- add_stripes_orthogonal_only
- add_stripes_over_row_extension
- add_stripes_partial_set_through_domain
- add_stripes_remove_floating_stapling
- add_stripes_remove_floating_stripe_over_block
- add_stripes_remove_stripe_under_ring
- add_stripes_respect_routes
- add_stripes_route_over_rows_only
- add_stripes_rows_without_stripes_only
- add_stripes_same_size_stack_vias
- add_stripes_skip_via_on_pin
- add_stripes_skip_via_on_wire_shape

- add_stripes_spacing_from_block
- add_stripes_spacing_type
- add_stripes_split_long_via
- add_stripes_split_vias
- add_stripes_split_wire_spacing
- add_stripes_split_wire_weight
- add_stripes_split_wire_width
- add_stripes_stacked_via_bottom_layer
- add_stripes_stacked_via_top_layer
- add_stripes_stapling_extend_to_minimum_spacing
- add_stripes_stapling_nets_style
- add_stripes_stapling_shift
- add_stripes_stop_at_closest_target
- add_stripes_stop_at_last_wire_for_area
- add_stripes_stripe_min_length
- add_stripes_stripe_min_width
- add_stripes_switch_cell_name
- add_stripes_switch_layer_overlap_length
- add_stripes_trim_antenna_back_to_shape
- add_stripes_trim_antenna_max_distance
- add_stripes_trim_stripe
- add_stripes_use_exact_spacing
- add_stripes_use_point_to_point_route
- add_stripes_use_stripe_width
- add_stripes_via_using_exact_crossover_size
- add_stripes_width_file

To set the value of any root attribute, use the `set_db` command:

```
set_db attribute_name value
```

To get the current value of any root attribute, use the `get_db` command:

```
get_db attribute_name
```

To get the current value of all attributes in the `add_stripes` category, use the following command:

```
get_db -category add_stripes
```

add_stripes_allow_jog

`add_stripes_allow_jog {none|{[padcore_ring ][block_ring ]}}`

Default: `padcore_ring block_ring`

Data_type: enum* {none padcore_ring block_ring}, read/write

Specifies the type that stripe can jog to connect.

If set to `padcore_ring` and/or `block_ring`, the stripes are allowed to be on the same/different metal layer to connect to `padcore_ring`/`block_ring`.

If set to `none`, the stripe is not allowed to jog on any layer. A straight/direct connection (no jog) with switch layer is allowed to make the connection.

add_stripes_allow_non_preferred_dir

```
add_stripes_allow_non_preferred_dir {none | blockring| corering | padring}
```

Default: `padring`

Data_type: enum* {none blockring corering padring stripe}, read/write

Specifies whether to allow non-preferred direction to jog to target. When set to `none`, the non-preferred direction jogging is disabled.

Note:

- `0` or `false` is not a legal value in this parameter.
- There are five available values in this parameter.
- The value `none` is exclusive to other option values.
- The default value is `padring`.

add_stripes_area

```
add_stripes_area string
```

Default: `""`

Data_type: string, read/write

Specifies the coordinates of the rectangular and/or rectilinear design area(s) in which the stripes must be created. It appends the specified area to previous ones.

Note:

- If `add_stripes_area` conflicts with `add_stripes -area`, `add_stripes -area` being on higher priority, `add_stripes_area` is ignored.
- If stripes are only to be created in the area in current command, reset the previous areas by one of the following three ways:

  ```
  set_db add_stripes_area -reset
  ```

  ```
  set_db add_stripes_area {}
  ```

  ```
  set_db add_stripes_area ""
  ```

- When this attribute is used with `add_stripes_domain_offset_from_core true`, the `add_stripes -start_offset` value for specified area is measured from core on using `add_stripes_area`.

add_stripes_blocks_without_same_net

```
add_stripes_blocks_without_same_net {none | block_ring | selected_block | unassigned_bump | overlap_ringpin | outer_ring
| outside_ringmacro | blocks_without_same_net}
```

Default: `none`

Data_type: enum* {none block_ring selected_block unassigned_bump overlap_ringpin outer_ring outside_ringmacro blocks_without_same_net}, read/write

Specifies the stripe break point.

add_stripes_break_at

```
add_stripes_break_at {none|{[block_ring] [selected_block] [unassigned_bump][overlap_ringpin] [outer_ring][outside_ringmacro]}}
```

Default: `none`

Data_type: enum* {none block_ring selected_block unassigned_bump overlap_ringpin outer_ring outside_ringmacro blocks_without_same_net}, read/write

Specifies the break point of stripes.

If any of the options `block_ring`, `selected_block`, `unassigned_bump`, `overlap_ringpin`, `outer_ring` or `outside_ringmacro` is specified, the stripe

breaks when it encounters the specified structure that are not in the same wire group.

`block_ring`
Stripes break and connect to the block rings and ring pins if they are encountered.

Note: This attribute applies only to block rings that surround a block, and does not apply to block rings that surround rows or standard cells. If the block
is a ring macro, the stripe will break at the ring pin inside the block ring.

`selected_block`
Stripes stop at the block boundary when the block does not contain a blockage.

Stripes stop at the block boundary when the block contains a blockage and spacing from the block boundary to the edge of the blockage is greater than the minimum spacing value of that layer.

Stripes stop at the minimum spacing away from the blockage when the spacing from the block boundary to the edge of the blockage is less than the minimum spacing of that layer.

Note: If the selected block has block ring, stripes break and connect to the block ring.

`unassigned_bumps`
Stripes break at unassigned bumps. It prevents the unassigned bumps from being ignored and shorted with the power and ground nets by the stripes created in the same metal layer.

`overlap_ringpin`
Prevents stripes from overlapping with same direction ring pins of a different net.

Note: This option is only valid if the ring pin is more than half the length of the macro in the direction of the ring pin.

`outer_ring`
When a stripe encounters nested block rings or ring pins that are not in the same wire group, the stripe breaks at the outer ring. If not specified, the stripe breaks at the innermost ring.

Note: This attribute works only in conjunction with the following attribute values of `add_stripes_break_at`:

- `block_ring`
- `selected_block`

`outside_ringmacro`
Stripes break at a small distance outside the block if the selected block has ring pins.

Note: This attribute is available only if you also specify `add_stripes_break_at selected_block`

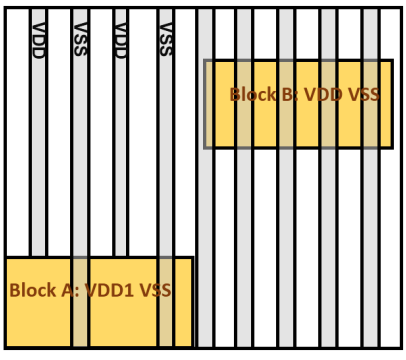
`blocks_without_same_net`
Breaks stripes at blocks, where no pin has same net as the nets specified in the `add_stripes` command.

Example

Block A has power supply VDD1 and VSS, Block B has power supply VDD and VSS, commands are as follows:

```
set_db add_stripes_break_at blocks_without_same_net
add_stripes -nets {VDD VSS}…
```

In the above commands, VDD breaks at Block A, VSS goes through Block A, VDD and VSS will go through Block B. Refer to the screen below:



add_stripes_continue_on_no_selection

`add_stripes_continue_on_no_selection {true | false}`

Default: `false`

Data_type: bool, read/write

If the design has no power domain, or no fences are selected, `add_stripes` ignores the `-over_power_domain` parameter and generates stripes extending to core ring.

add_stripes_detailed_log

`add_stripes_detailed_log {true | false}`

Default: `false`

Data_type: bool, read/write

Setting this attribute will make the `add_stripes` output an extensive log or a detailed log.

add_stripes_domain_offset_from_core

`add_stripes_domain_offset_from_core {true | false}`

Default: `false`

Data_type: bool, read/write

Creates stripes over a power domain to align with the global stripes in your design by enabling you to use the offset that was used for the global stripes as the offset for the power domain stripes. This attribute uses the offset from the core, which is the default for global stripes.

add_stripes_extend_to_closest_target

`add_stripes_extend_to_closest_target {none | stripe | ring | same_dir_stripe | area_boundary}`

Default: `none`

Data_type: enum* {ring stripe same_dir_stripe area_boundary none}, read/write

Extends the stripe to the specified target. This attribute can work in conjunction with `add_stripes_max_extension_distance`.

Following is the description of the attribute values:

- `none`: Restricts the stripe from extending to any target.
- `stripe`: Allows the stripe to extend to the nearby stripe.
- `ring`: Allows the stripe to extend to the nearby ring.
- `same_dir_stripe`: Allows the stripe to extend to the stripe in the same direction.
- `area_boundary`: Allows the stripe to extend to certain area boundary.

add_stripes_extend_to_first_ring

`add_stripes_extend_to_first_ring {true | false}`

Default: `false`

Data_type: bool, read/write

If set to `true`, stripe antennas are extended to the closest ring. If set to `false`, stripes stop at the boundary specified by the `-extend_to` attribute of the `add_stripes` command.

add_stripes_ignore_block_check

`add_stripes_ignore_block_check {true | false}`

Default: `false`

Data_type: bool, read/write

If set to `false`, when a stripe encounters a ring, the software does not check whether that ring is surrounding a block. If set to `true`, when a stripe encounters a ring, the software automatically checks whether a block is enclosed within the ring.

Note: This attribute only works in conjunction with the `-break_at` attribute of the `add_stripes` command.

add_stripes_ignore_block_ring_during_break

`add_stripes_ignore_block_ring_during_break {true | false}`

Default: `false`

Data_type: bool, read/write

If set to `false`, breaks the stripe outside the block if the stripe cannot go over the block.

If set to `true`, breaks the stripe at the block ring if it cannot go over the block.

add_stripes_ignore_drc

`add_stripes_ignore_drc {true | false}`

Default: `false`

Data_type: bool, read/write

Setting this attribute creates stripes ignoring DRCs but honoring user defined values. This attribute should be used only for prototyping.

add_stripes_ignore_non_default_domains

`add_stripes_ignore_non_default_domains {true | false}`

Default: `false`

Data_type: bool, read/write

Creates global stripes over domains without breaking. If you do not want the stripes to break at the non-default domains, set this variable to `true` before using the `add_stripes` command.

add_stripes_in_cell_allow_shift

`add_stripes_in_cell_allow_shift {true | false}`

Default: `false`

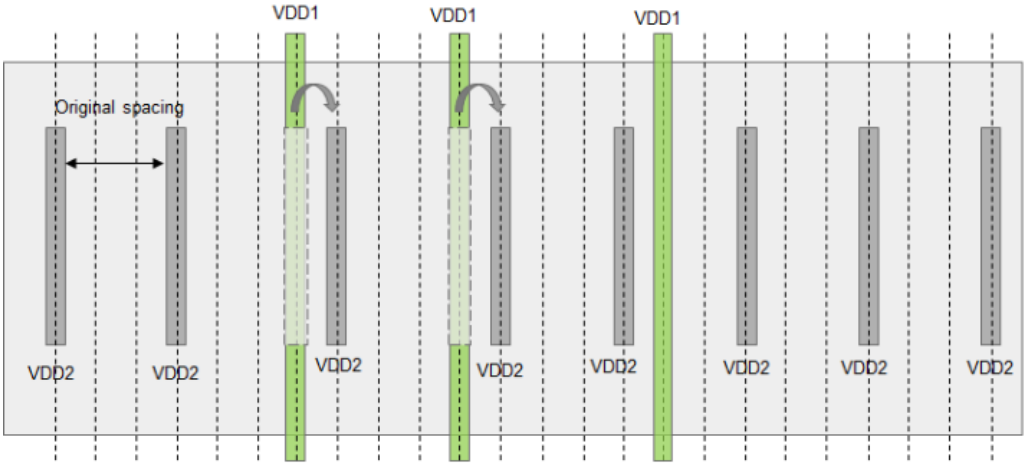
Data_type: bool, read/write

Allows to shift the newly created pillars when meeting DRC, if there is already an existing `special_wire` on the given track. This attribute works only when the `add_stripes_in_cell_only` attribute is set to `true`.

When the `add_stripes_in_cell_allow_shift` attribute is set to `false`, the pillar is removed when meeting violation. When set to `true`, the newly created pillar is shifted to right among the three tracks when meeting violation. The next pillar's position depends on the final position of the current pillar.

The distance between these pillars should satisfy the user specified spacing. The spacing between the two pillars should not be smaller than the user-specified spacing.

The following example shows that using the `add_stripes_in_cell_allow_shift` attribute, the newly created pillars of `VDD2` are shifted right to avoid DRC with `VDD1`.



add_stripes_in_cell_only

`add_stripes_in_cell_only {true | false}`

Default: `false`

Data_type: bool, read/write

Control whether the stripe will be generated inside selected or specified instances. If no instance is selected or without the `add_stripes -master` parameter, no stripe will be generated.

add_stripes_keep_pitch_after_snap

`add_stripes_keep_pitch_after_snap {true | false}`

Default: `false`

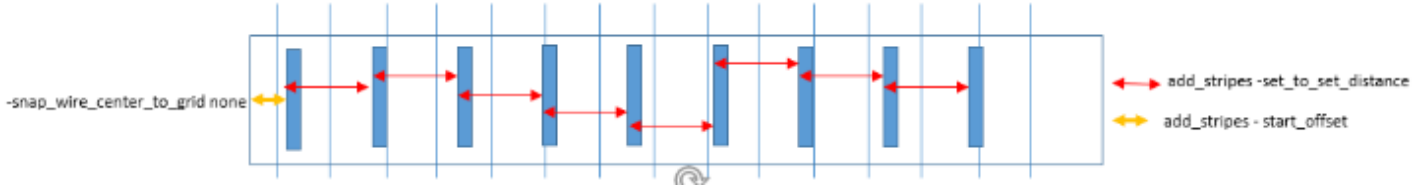
Data_type: bool, read/write

When set to `true`, the distance between two PG sets for snapping is set to greater than or equal to the pitch value specified through the `add_stripes -set_to_set_distance` parameter. When set to `false`, the newly created PG sets are snapped from the current location to the nearest location valid for the `add_stripes_keep_pitch_after_snap` attribute. The average pitch of the newly generated stripe is approximately equal to the specified `add_stripes -set_to_set_distance` parameter value.

This attribute works with the `add_stripes -snap_wire_center_to_grid` parameter. The behavior of the `add_stripes_keep_pitch_after_snap` attribute under different scenarios is explained below.

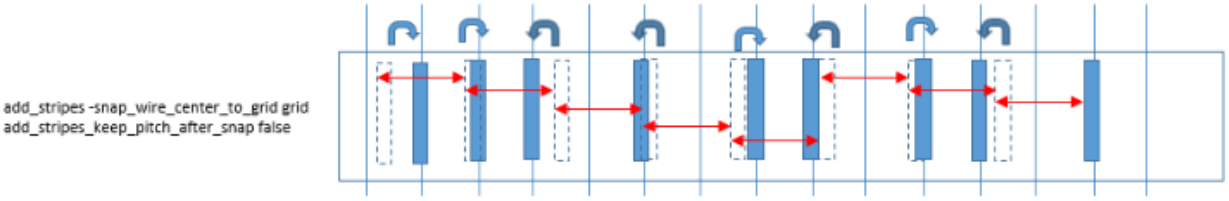
Scenario 1:

When the `add_stripes -snap_wire_center_to_grid` parameter is set to `none`, the `add_stripes_keep_pitch_after_snap` attribute does not work. See the example below for understanding:



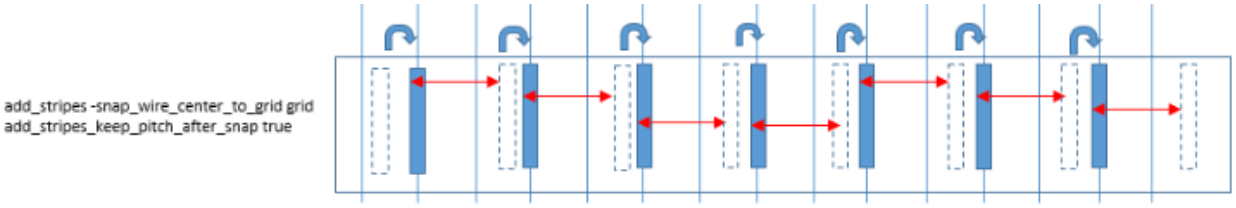
Scenario 2:

When the `add_stripes -snap_wire_center_to_grid` parameter is set to `grid`, and the `add_stripes_keep_pitch_after_snap` attribute is set to `false`, all the locations for snapping are decided based on the start location. Post then, the stripes are snapped to the nearest track. The average distance between of newly generated stripes is approximately equal to the specified pitch (`add_stripes -set_to_set_distance` parameter value). See the example below for understanding:



Scenario 3:

When the `add_stripes -snap_wire_center_to_grid` parameter is set to `grid`, and the `add_stripes_keep_pitch_after_snap` attribute is set to `true`, the location for snapping is decided based on the previous location. Post then, the stripes are snapped to the next track. See the example below for understanding:



Note:

- The first stripe of the design always snaps to the next track to meet the `add_stripes -start_offset` value. To start the first stripe snapping at the nearest track, set the `add_stripes -allow_snapping_override_custom_spacing` parameter to `1`.
- By default, the `add_stripes_keep_pitch_after_snap` attribute is turned on only at the `N10` nodes.

add_stripes_mask_color_balance

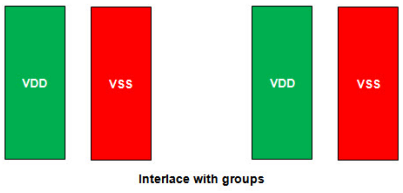
`add_stripes_mask_color_balance {same_color | alternate_color | none}`

Default: `none`

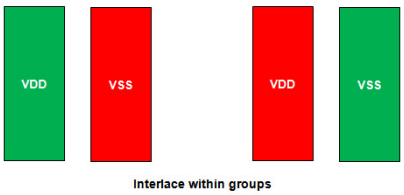
Data_type: enum {same_color alternate_color none}, read/write

Specifies addition of VDD/VSS stripes with balanced color.

`same_color`
Corresponds to interlace with groups.



`alternate_color`
Corresponds to interlace within the group.



To achieve color balancing, the color of stripes occupying even number tracks is picked alternatively.

Note: The stripes location will always be either on full grid/track or half grid/track:

- If the stripes occupy odd number of tracks, they must be on full grid.

- If the stripes occupy even number of tracks, they must be on half grid.

To achieve color balance, the stripes color will be as mentioned below:

- If the stripes occupy odd number of tracks, the color is opposite to the center track color.
- If the stripes occupy even number of tracks, the color can be any of red or green and will keep alternating.

Only two nets are supported.

The attribute must be specified along with `add_stripes_width_file`.

add_stripes_max_extension_distance

`add_stripes_max_extension_distance` *`distance`*

Default: `2.14748e+09`

Data_type: double, read/write

Sets the maximum extension distance between stripes to nearby stripes in the same layer and the same direction.

add_stripes_max_via_size

`add_stripes_max_via_size {shape width% height% target_penetration%}`

Default: `""`

Data_type: string, read/write

The maximum size of a crossover via, specified as a percentage of a full-size via. You must specify three values for each shape that you specify.

The shape values are:

- Pad
  via size between stripe and pad pin/ring
- Ring
  via size between stripe and core ring
- Stripe
  via size between stripes
- Blockring
  via size between stripe and block ring
- Blockpin
  via size between stripe and block pin, including ring pin
- Cover
  via size between stripe and the macro which has CLASS COVER defined in LEF
- Selected
  via size between stripe and selected objects
- No shape
  via size between stripe and wires with shape None
- Standardcell
  via size between stripe and standarcell pin

The three values you must specify for each shape are:

- The maximum width for the crossover via, as a percentage of a full-size via
- The maximum height for the crossover via, as a percentage of a full-size via
- The target penetration, which specifies a smaller size for vias connecting to targets of the specified shape

add_stripes_merge_with_all_layers

```
add_stripes_merge_with_all_layers {true | false}
```

Default: `false`

Data_type: bool, read/write

Merges a stripe to a block ring or pin on all layers. This attribute is similar to the `-merge_stripes_value` attribute of the `add_stripes` command, except that it merges a stripe to all layers instead of only the stripe layer.

add_stripes_mesh_via

```
add_stripes_mesh_via  {true | false}
```

Default: `false`

Data_type: bool, read/write

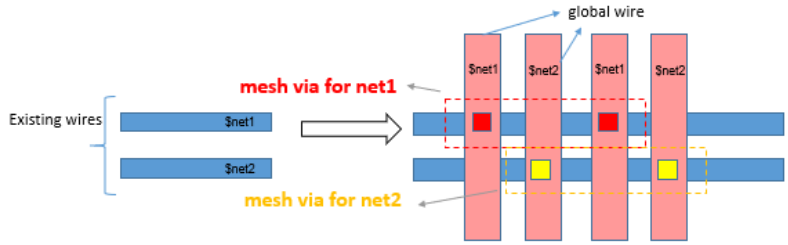
Enables to use mesh via pillars to reduce memory.

Example 1:

The following commands use mesh via pillars with the help of the `-mesh_via` parameter:

```
set_db add_stripes_mesh_via 1
add_stripes -nets {$net1 $net2 $net1 $net2} ...
```

Following is the output:

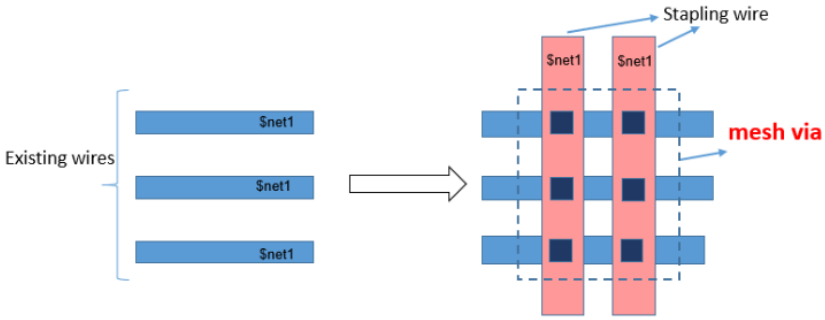


Example 2:

The following commands generate mesh via automatically in one stapling stripe:

```
set_db add_stripes_mesh_via 1
add_stripes -stapling{ …} -nets {$net1 …$net1…} …
```

Following is the output:



add_stripes_offset_from_core

```
add_stripes_offset_from_core {true | false}
```

Default: `true`

Data_type: bool, read/write

If set to `true`, the first or last stripe is offset from the core area of the design. If set to `0`, the first or last stripe is offset from core ring, pad ring, or design boundary, depending on the attributes specified with the `add_stripes` command.

Note: This attribute does not affect the offset if you specify the add_stripes command with any of the following attributes:

- `-over_power_domain 1`
- `-extend_to all_domains`
- `-area`

add_stripes_opt_stripe_for_routing_track

```
add_stripes_opt_stripe_for_routing_track {shift | none}
```

Default: `none`

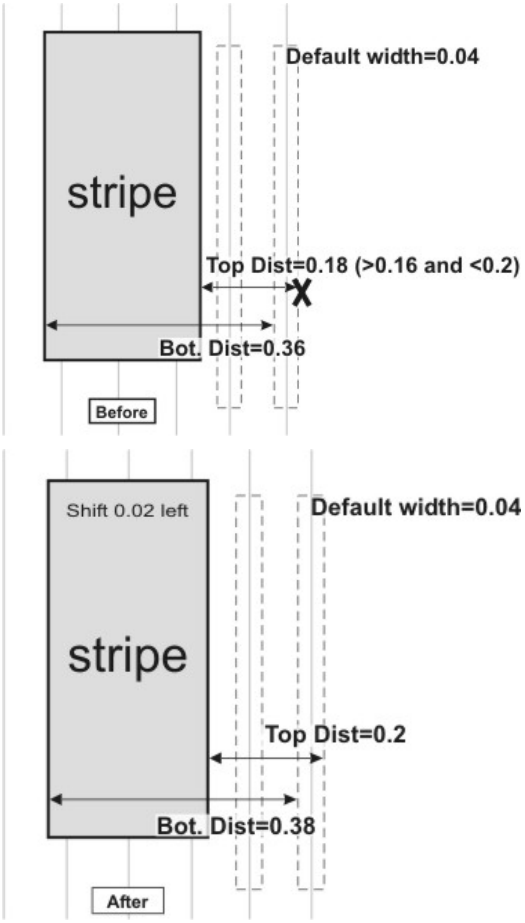
Data_type: enum* {shift none}, read/write

Specifies the location setting to move stripe center line to avoid the forbidden spacing. If both sides cannot be fixed at the same time, one side is always fixed.

When set to this attribute to the `shift` value, `add_stripes` automatically shifts stripes to preserve routing resource.

Example

For the rule below,
FORBIDDENSPACING 0.16 0.2 WIDTH 0.05 PRL 0;



After shifting the stripe 0.02 left, Top Dist. and Bot. Dist. are >=0.2, and one track is saved.

add_stripes_orthogonal_offset

```
add_stripes_orthogonal_offset {[{edge1 orthogonal_offset1} {edge2 edge3 orthogonal_offset2} {edge4 edge5 … orthogonal_offset3 }
…]|[all orthogonal_offset]}
```

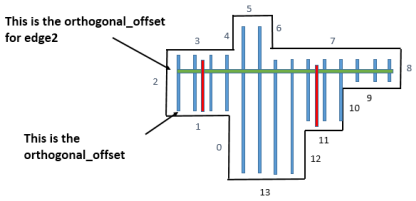
Default: `{{none}}`

Data_type: string, read/write

Defines `orthogonal_offset` of stripes for every edge of block level designs, blocks or standard cells for specifying the offsets for edges orthogonally. Edge number starts from the left bottom vertical edge. When this attribute is set for the second time, all the `orthogonal_offset` values are reset firstly.

Example 1:

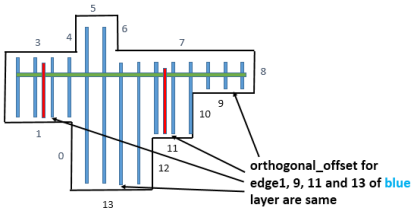
```
set_db add_stripes_orthogonal_offset {{1 orthogonal_offset1} {2 orthogonal_offset2} {3 orthogonal_offset3} {5 orthogonal_offset5}
{7 orthogonal_offset7} {8 orthogonal_offset8} {9 orthogonal_offset9} {11 orthogonal_offset11} {13 orthogonal_offset13}}
add_stripes -layer green …
add_stripes -layer blue …
set_db add_stripes_orthogonal_offset {{1 orthogonal_offset1} {3 orthogonal_offset3} {7 orthogonal_offset7} {11
orthogonal_offset11}}
add_stripes -layer red …
```



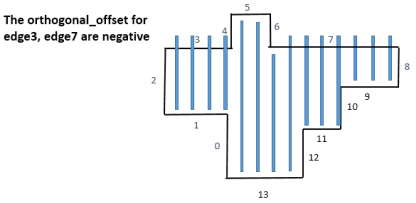
Example 2:

The following commands ensure that the stripes are generated in the block with a specified orthogonal offset:

```
set_db add_stripes_orthogonal_offset {{1 9 11 13 orthogonal_offset1} {2 orthogonal_offset2} {3 orthogonal_offset3} {5
orthogonal_offset5} {7 orthogonal_offset7} {8 orthogonal_offset8}}
add_stripes -layer green …
add_stripes -layer blue …
set_db add_stripes_orthogonal_offset {{1 orthogonal_offset1} {3 orthogonal_offset3} {7 orthogonal_offset7} {11
orthogonal_offset11}}
add_stripes -layer red …
```




```
set_db add_stripes_orthogonal_offset {{3 7 orthogonal_offset3} {1 5 9 11 13 orthogonal1}}
add_stripes -layer blue …
```



add_stripes_orthogonal_only

```
add_stripes_orthogonal_only {true | false}
```

Default: If you do not specify this attribute or if you specify it with a value of `1`, the software creates vias for orthogonal wires and pin intersections.

Data_type: bool, read/write

Creates vias *only* for orthogonal wires and pin intersections.

add_stripes_over_row_extension

```
add_stripes_over_row_extension {true | false}
```

Default: `false`

Data_type: bool, read/write

Extends the stripes to cover the followpin at the row end.

add_stripes_partial_set_through_domain

```
add_stripes_partial_set_through_domain {true | false}
```

Default: `false`

Data_type: bool, read/write

Specifies whether to go over the power domain if domain contains the specified net.

add_stripes_remove_floating_stapling

```
add_stripes_remove_floating_stapling {true | false}
```

Default: `true`

Data_type: bool, read/write

Removes those floating stapling stripes that are not connected to the power stripe/follow pin, when the stapling stripes are generated using a reference layer. This attribute does not remove floating stapling stripes when they are added through the user-specified offset and pitch.

add_stripes_remove_floating_stripe_over_block

```
add_stripes_remove_floating_stripe_over_block {true | false}
```

Default: `true`

Data_type: bool, read/write

If set to `true`, stripe fragments that start and end inside the same block or ring macro are removed. If set to `0`, stripe fragments are not removed.

add_stripes_remove_stripe_under_ring

```
add_stripes_remove_stripe_under_ring {true | false}
```

Default: `false`

Data_type: bool, read/write

Removes stripe segments, which have overlapped area and are in the same direction as the existing ring. The stripe can be in a different layer to the existing ring.

add_stripes_respect_routes

```
add_stripes_respect_routes {none | fixed | fixed_and_clock | all}
```

Default: `fixed_and_clock`

Data_type: enum, read/write

Specifies the route types to ignore DRC check during stripe generation. This attribute always honor all covers and PGs, but does not honor the trail routes, Following are the attribute values:

- `none`: Specifies that no route types are respected.
- `fixed`: Respects fixed regular wires.
- `fixed_and_clock`: Respects fixed and routed clocks.
- `all`: Respects fixed and routed clocks along with the routed signals.

The following table presents the support of the `add_stripes_respect_routes` attribute values with various statuses of net:

Target PG Honoring a Wire			add_stripes_respect_routes Values			
Net Use	**Status**		**none**	**fixed**	**fixed_and_clock**	**all**
P/G			Yes	Yes	Yes	Yes
Non-P/G	**Cover**		Yes	Yes	Yes	Yes
	Fixed		No	Yes	Yes	Yes
	Routed	**Clock**	No	No	Yes	Yes
		Signal	No	No	No	Yes
	Unknown		No	No	No	No

add_stripes_route_over_rows_only

```
add_stripes_route_over_rows_only {true | false}
```

Default: `false`

Data_type: bool, read/write

If set to `true`, route stripes only over standard cells rows. If set to `0`, stripes are not limited to being routed over rows within this boundary.

add_stripes_rows_without_stripes_only

```
add_stripes_rows_without_stripes_only {true | false}
```

Default: `false`

Data_type: bool, read/write

If set to `true`, global stripes are generated as specified by other attributes. Then stripes are also generated over row clusters that do not already contain stripes. If set to `false`, stripes are generated as specified by other attributes and not over row clusters without stripes.

add_stripes_same_size_stack_vias

```
add_stripes_same_size_stack_vias {true | false}
```

Default: `false`

Data_type: bool, read/write

If set to `true`, the software always generates same-sized stacked vias. If set to `false`, the software trims the vias as needed to meet the DRC rules.

add_stripes_skip_via_on_pin

```
add_stripes_skip_via_on_pin {[Pad] [Block] [Cover] [Standardcell] [Physicalpin]}
```

Default: `Standardcell`

Data_type: enum* {pad block cover standardcell physicalpin}, read/write

Prevents vias from being generated on the specified types of pins.

If you specify `-skip_via_on_pin {}`, the power planning software makes connections to all five categories of pins. If you do not specify this attribute, vias are generated for block pins, pad pins, and cover macro pins, but are not generated for standard cell pins.

add_stripes_skip_via_on_wire_shape

```
add_stripes_skip_via_on_wire_shape { [Blockring] [Stripe] [Padring] [Ring] [Noshape] }
```

Default: If you do not specify this attribute, vias can be generated on all wire shapes *except* no-shape wires.

Data_type: enum* {blockring stripe followpin corewire blockwire iowire padring ring fillwire noshape}, read/write

Prevents vias from being generated on the specified wire shapes.

If you specify `-skip_via_on_wire_shape {}`, the power planning software makes connections to all five categories of wire shapes.

add_stripes_spacing_from_block

```
add_stripes_spacing_from_block float
```

Default: `0`

Data_type: double, read/write

If the value is less than 0, over-the-block stripes end the specified distance from the boundary.

If the value is greater than 0, global stripes break at the specified user units from the obstructed blocks.

If the value is equal to 0, the stripe offset is 0.

add_stripes_spacing_type

```
add_stripes_spacing_type {enum {edge_to_edge | center_to_center}}
```

Default: `edge_to_edge`

Data_type: enum {edge_to_edge center_to_center}, read/write

Specifies the edge-to-edge spacing between stripes in each set.

add_stripes_split_long_via

```
add_stripes_split_long_via { threshold_value step_value offset_value height_value }
```

Default: If you do not specify this attribute, the software does not split vias.

Data_type: string, read/write

Splits vias longer than the specified length (`threshold_value`, in micrometers) into smaller vias with specified center-to-center spacing (`step_value`, in micrometers), left/bottom end offset (`offset_value`, in micrometers) and vertical/horizontal height (`height_value`, in micrometers) values . This is independent of the `add_stripes_split_vias` attribute.

Note: If *`offset_value`* is specified as negative, the vias will be placed symmetrically on both ends. The default value of `offset_value` is `-1` .

add_stripes_split_vias

```
add_stripes_split_vias {true | false}
```

Default: `true`

Data_type: bool, read/write

Specifies whether two or more partial vias can be created at the crossover between a stripe and any target (pin or wire) if an obstruction (macro

obstruction; wire or pin on a different net) prevents the creation of a full-size via. This can be useful, for example, if several metal blockages partially obstruct the area where a full-size via would normally be placed. This attribute permits the creation of partial vias that would not violate the process rules, in the open areas. A value of `1` allows multiple partial vias to be created. A value of `0` prevents multiple partial vias from being created.

add_stripes_split_wire_spacing

`add_stripes_split_wire_spacing` *`floating_point_value`*

Default: If you omit this attribute, or set its value to `0`, the software uses the spacing rule in the LEF file.

Data_type: double, read/write

Specifies the spacing between each split wire.

add_stripes_split_wire_weight

`add_stripes_split_wire_weight` *`integer_0_to_10`*

Default: If you omit this attribute, or set its value to `0`, no I/O wires are deleted when the above connections are made.

Data_type: double, read/write

When split wires from a stripe on one layer attempt to make a target connection on another layer but are obstructed by I/O wires, some I/O wires might be deleted so that approximately *n* of 10 split stripe wires complete the target connection.

Note: When you specify this option, the software deletes the fewest I/O wires to accommodate approximately *n* of 10 split stripe wires.

add_stripes_split_wire_width

`add_stripes_split_wire_width` *`floating_point_value`*

Default: If you omit this attribute, or set its value to `0`, the software uses the maximum width value in the LEF file.

Data_type: double, read/write

Specifies the width of each split wire.

add_stripes_stacked_via_bottom_layer

`add_stripes_stacked_via_bottom_layer` *`layername`*

Default: The lowest metal layer in the design.

Data_type: string, read/write

Specifies the lowest layer in which vias can be stacked.

add_stripes_stacked_via_top_layer

`add_stripes_stacked_via_top_layer` *`layername`*

Default: The highest metal layer in the design.

Data_type: string, read/write

Specifies the highest layer in which vias can be stacked.

add_stripes_stapling_extend_to_minimum_spacing

`add_stripes_stapling_extend_to_minimum_spacing {true | false}`

Default: `false`

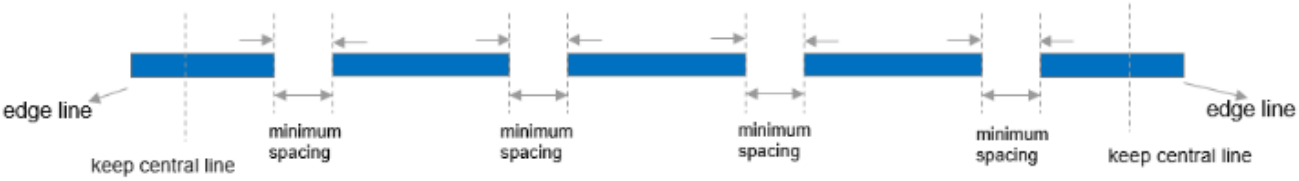
Data_type: bool, read/write

Automatically extends the newly created adjacent stapling to satisfy the end-to-end minimum spacing requirement, which will also be free

from DRCs. This attribute works only with the stapling wires.

When set to to `false`, the length of the newly added stripe follows the length defined through the `add_stripes -stapling` parameter. The following figure presents the placing of stripes when this attribute is set to `false`:

When set to `true`, the newly added stripes are extended automatically. The wire on the edge with only one end will have end-to-end spacing with the newly generated stripes. Only the end-to-end side is extended and the edge line end is not extended. The extended stripe still remains within the die area. The following figure presents the placing of the extended stripes when this attribute is set to `true`:



add_stripes_stapling_nets_style

```
add_stripes_stapling_nets_style {end_to_end | side_to_side | side_to_side_full_nets}
```

Default: `end_to_end`

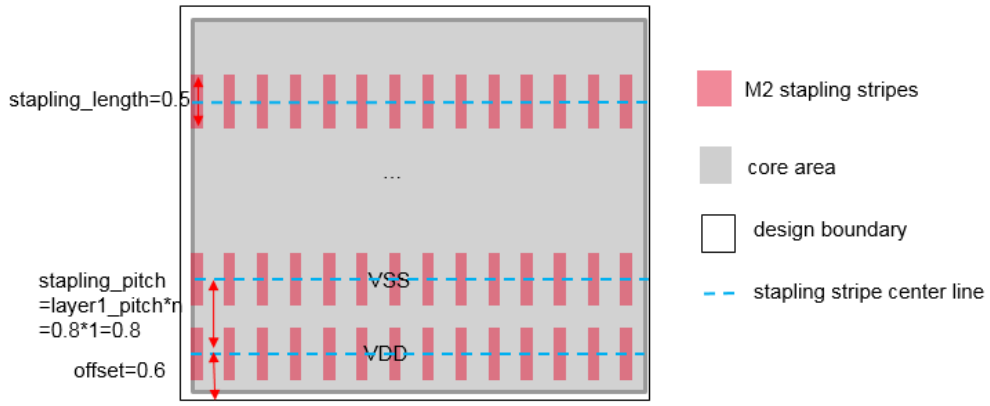
Data_type: enum {end_to_end side_to_side side_to_side_full_nets}, read/write

Specifies the stapling style, if multiple nets are used. Please note that:

- When `end_to_end` (default value) is specified, nets are arranged in end direction. Refer to the following examples:

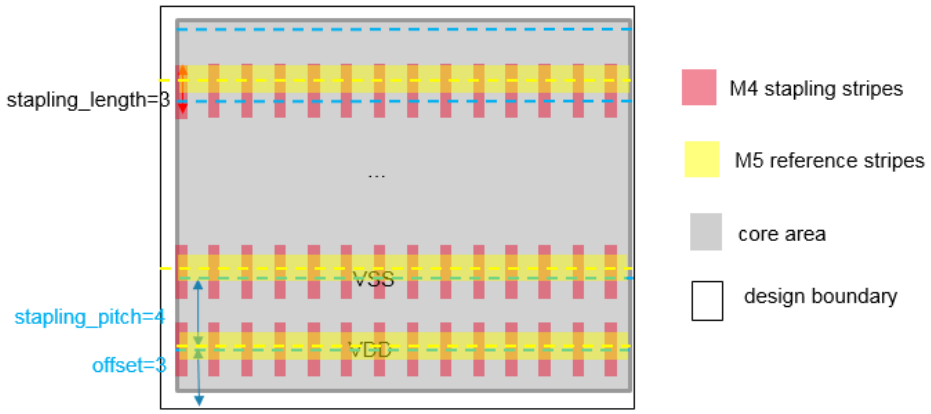
  Example 1

  ```
  add_stripes -stapling {0.5 0.6 0.8:1} -nets {VDD VSS} -layer metal2 -width 0.1 -direction vertical -set_to_set_distance 0.5
  ```

  Example 2

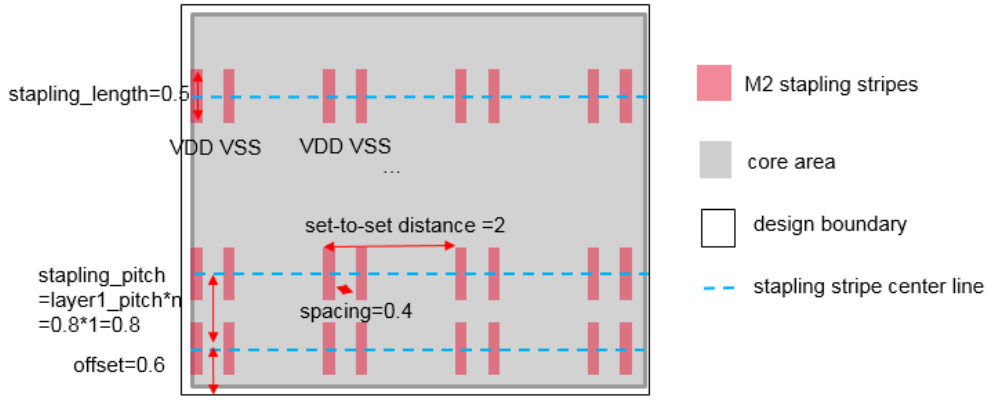
  ```
  set metal3_track_offset 3
  set metal3_pitch 0.8
  add_stripes -stapling '3 $metal3_track_offset $metal3_pitch:5 metal5' -nets {VDD VSS} -layer metal4 -width 0.5 -direction
  vertical -set_to_set_distance 2
  ```



- When `side_to_side` is specified, nets are arranged in the side direction. Refer to the following example:

  Example 3

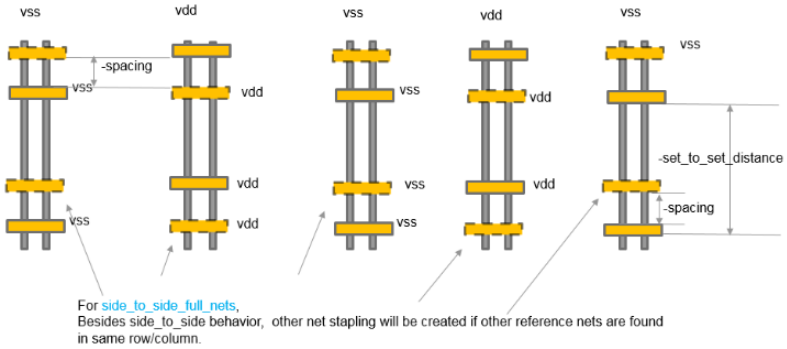
  ```
  add_stripes -stapling {0.5 0.6 0.8:1} -nets {VDD VSS} -spacing 0.4 -layer metal2 -width 0.1 -direction vertical -
  set_to_set_distance 2
  ```



- When `side_to_side_full_nets` is specified, nets are arranged in the side direction. Besides the `side_to_side` behavior, it creates other net stapling according to the reference nets. The `side_to_side_full_nets` value can be generated only over the existing reference stripe (with `-stapling {length refer_layer}`). Refer to the following example:

  Example 4

  ```
  -stapling_nets_style side_to_side_full_nets #Besides side_to_side behavior, creating other net stapling according to the
  reference nets
  -stapling {length refer_layer} #Stapling stripe is generated over existing reference stripe
  ```



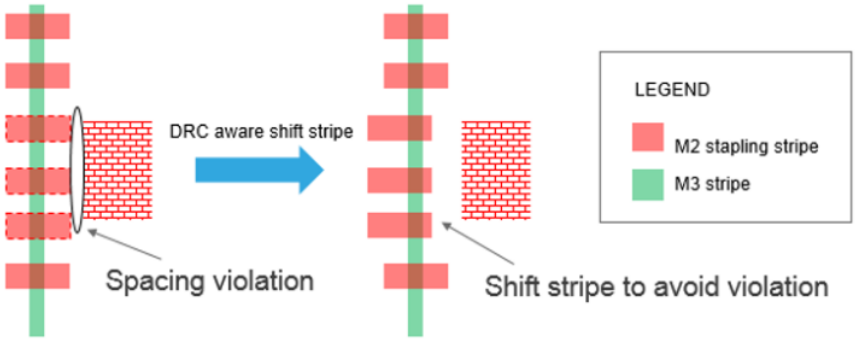
add_stripes_stapling_shift

`add_stripes_stapling_shift {true | false}`

Default: `true`

Data_type: bool, read/write

Shifts the stapling stripes automatically to avoid the DRC. See the example below:



add_stripes_stop_at_closest_target

`add_stripes_stop_at_closest_target { none | block_ring | core_ring | stripe }`

Default: `none`

Data_type: enum* {none block_ring core_ring stripe}, read/write

Stops the stripe at closest target while drawing the stripe on the master cell.

add_stripes_stop_at_last_wire_for_area

`add_stripes_stop_at_last_wire_for_area {true | false}`

Default: `false`

Data_type: bool, read/write

If set to `false`, the stripe antennas are trimmed back to the closest wire or pin of the same net. If set to `true`, stripe antennas are not trimmed.

Note: This option is only available if you specify the `-area` parameter of the `add_stripes` command.

add_stripes_stripe_min_length

`add_stripes_stripe_min_length {floating_point_value | stripe_width}`

Default: `stripe_width`

Data_type: string, read/write

Specifies the minimum stripe length. `stripe_width` is a fixed string. The fixed string here refers to the width of present wire.

add_stripes_stripe_min_width

`add_stripes_stripe_min_width floating_point_value`

Default: `0.0`

Data_type: double, read/write

Specifies the minimum width of the remaining stripe after being trimmed.

add_stripes_switch_cell_name

`add_stripes_switch_cell_name { cellA cellB...}`

Default: `none`

Data_type: string, read/write

Treats the specified switch cell as a ring. Pins in the switch cell can then be used as targets for stripes. If the switch cell is in the core area, the software treats it as a block ring. If the switch cell is outside of the floorplan row, the cell is treated as a core ring, and used as the stripe boundary.

You can specify one or more switch cell names.

To turn off this functionality, specify one of the following:

- `-switch_cellname ""`
- `-switch_cellname none`

add_stripes_switch_layer_overlap_length

`add_stripes_switch_layer_overlap_length floating_point_value`

Default: If you omit this attribute, or set its value to `0`, the power planning software creates a square via.

Data_type: double, read/write

Controls the overlap distance used for the via between a stripe layer and the layer being switched to.

Note: The overlap applies *only* to vias created for straight (non-jogging) connections.

add_stripes_trim_antenna_back_to_shape

`add_stripes_trim_antenna_back_to_shape {none | block_ring | core_ring | pad_ring | stripe}`

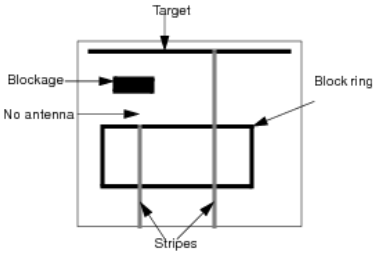
Default: `none`

Data_type: enum* {none block_ring core_ring pad_ring stripe}, read/write

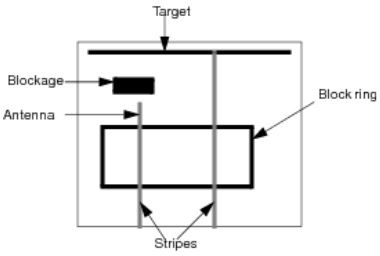
Prevents stripes from creating antennas when a wire of the specified shape is the last target before a blockage that prevents connection to another target. Specify one of the following shapes:

- `block_ring`
- `core_ring`
- `pad_ring`
- `stripe`

The following figure illustrates the software behavior if you specify this attribute with a value of `block_ring`.



Default: Stripes extend to the blockage, leaving an antenna between the last connection and the blockage. The following figure illustrates the software behavior if you do not specify this attribute.



add_stripes_trim_antenna_max_distance

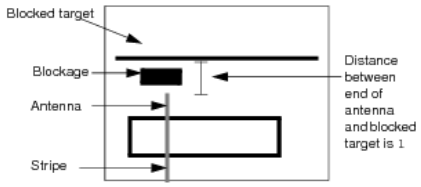
`add_stripes_trim_antenna_max_distance` *`distance`*

Default: `0.0`

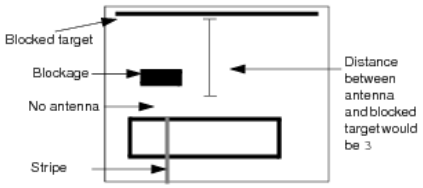
Data_type: double, read/write

Use this option to specify a distance, in micrometers, between the end of the antenna and a blocked target. This value controls whether an antenna is kept or trimmed for the selected shape. Cadence recommends that you specify a value equivalent to the distance between the core area and the pad ring or core ring.

- If the blocked target is within the specified distance, the antenna is kept, and you can use `route_special` to complete the connection. For example, if you specify a distance of `2` micrometers and the distance between the end of the antenna and the blocked target is `1` micrometer, the antenna is not trimmed.



- If the blocked target is beyond the specified distance, the antenna is trimmed back to the last connection. For example, if you specify a distance of `2` micrometers and the distance between the antenna and the blocked target would be `3` micrometers, the antenna is trimmed back.



add_stripes_trim_stripe

`add_stripes_trim_stripe { none |design_boundary | core_boundary }`

Default: `none`

Data_type: enum {none design_boundary core_boundary}, read/write

Specifies if shape is trimmed by the specified shape.

add_stripes_use_exact_spacing

`add_stripes_use_exact_spacing {true | false}`

Default: `false`

Data_type: bool, read/write

If set to `true`, the power stripe breaks using the exact spacing rule from the obstructing blockage, pin, or wire to avoid a design rule violation. If set

to `0`, the power stripe breaks at 1.5 times the maximum range rule spacing of all layers.

add_stripes_use_point_to_point_route

`add_stripes_use_point_to_point_route {true | false}`

Default: `false`

Data_type: bool, read/write

If set to `false`, the point-to-point router engine is turned off. If set to `true`, the `add_stripes` command automatically calls the internal point-to-point router engine to route the difficult power stripe connections (such as connecting power stripe antennas that were not connected by the `add_stripes` command).

add_stripes_use_stripe_width

`add_stripes_use_stripe_width {true | false}`

Default: `true`

Data_type: bool, read/write

If set to `true`, stripes that jog to connect to a ring always use the stripe width for the jog. If set to `false`, the jog is either the width of the stripe or the width of the ring segment, which ever is smaller.

Note: This attribute only works in conjunction with the `-jog_to_connect 0` parameter of the `add_stripes` command.

add_stripes_via_using_exact_crossover_size

`add_stripes_via_using_exact_crossover_size {true | false}`

Default: `true`

Data_type: bool, read/write

Specifies whether to generate partial vias of the exact crossover size. A value of `1` allows partial vias to be generated. A value of `0` prevents partial vias, and the software always generates full-size vias.

Note: This option does not apply when both of the objects at the crossover are rings or stripes, or at the crossover of a ring and stripe. In these cases, the software always generates a full-size via array.

add_stripes_width_file

`add_stripes_width_file` *`file_name`*

Default: `""`

Data_type: string, read/write

The `file_name` specifies a list of recommended width values for the stripes.

Note: This option must be specified along with `-color_balance`.

add_target_pg

```
  add_target_pg
[-help]
[-area {x1 y1 x2 y2}{x1 y1 x2 y2 x3 y3 x4 y4 ...}]
[-force]
[-instance {list_of_instances}]
-nets {list_of_nets}
-pin_layer layer
[-stack_layer_config {stack_layer_pattern}]
[-stack_layer_config_file filename]
-target_layer {layer1 layer2}
[-user_class subclass_string]
```

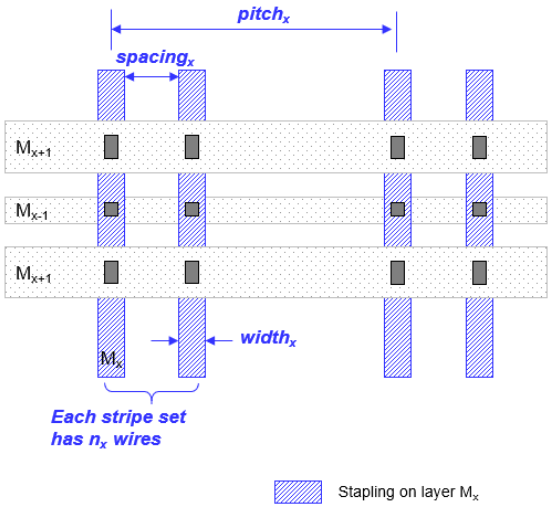
Creates a PG path for a certain cell or instance. There are several cases where the IR drop hotspots occur only on certain cells, like high driving strength clock buffer cell, and there are some cells wherein secondary PG pins need to be connected with power mesh. So, instead of increasing the density of regular PG mesh, the `add_target_pg` command allocates a dedicated PG path from the top layer low resistance PG stripe to these cells and solves most of the IR-drop hotspot issues. As a result, more resources can be preserved for routing.

Parameters

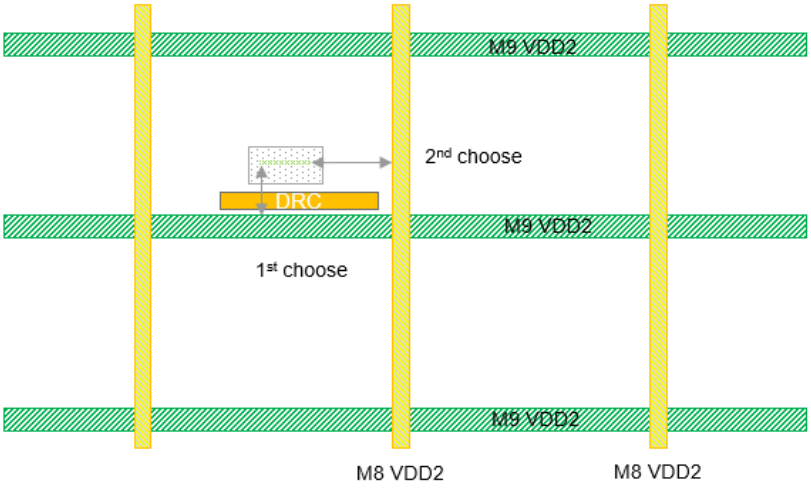
`-help`	Outputs a brief description that includes the type and default information for each `add_target_pg` parameter. For a detailed description of the command and all its parameters, use the `man` command: `man add_target_pg`
`-area {x1 y1 x2 y2}{x1 y1 x2 y2 x3 y3 x4 y4 ...}`	
	Signifies that an instance inside the specified area can be connected. *Data_type*: string, optional
`-force`	Keeps the added PG even when the connection has a DRC, and adds a DRC marker. When this parameter is not set, the newly added PG will remain DRC clean. In such a case, if the PG pin cannot connect to the target layer with DRC clean, the added PG is deleted and leaves the pin open. *Data_type*: bool, optional
`-instance {list_of_instances}`	Specifies the instances to be connected. Note that only those instances are connected, which belong to the cell defined by the `add_target_pg_pins` attribute. *Data_type*: string, optional
`-nets {list_of_nets}`	Specifies the power/ground nets to be connected. *Data_type*: string, optional
`-pin_layer layer`	Specifies the layer name of the cell pin to be connected. *Data_type*: string, optional
`-stack_layer_config {stack_layer_pattern}`	

	Defines the pattern of the stripes on intermediate layers between the `pin` layer and target layer. The value of this parameter is in the `{layer1:n1:s1:w1:p1 layer2:n2:s2:w2:p2 …}` format. Here: • `layer1`, `layer2`, … specify the layer names. • `n1`, `n2`, … specify the stapling count in a set. • `w1`, `w2`, … specify the stapling width. • `s1`, `s2`, … specify the spacing inside a stapling set. • `p1`, `p2`, … specify the pitch (set-to-set distance) between the stapling sets. Following is an example of the `-stack_layer_config` parameter value: `-stack_layer_config {M3:2:0.12:0.04:0.36 M4:1:0.08:0.08:0.9}` *Data_type*: string, optional

`-stack_layer_config_file` *filename*	Specifies the stack layer configuration file defining the pattern of the stripes on intermediate layer between the pin and target layers. Stack layer denotes the intermediate layer(s) between the cell pin and target stripe. Following is the syntax of the stack layer configuration file:
	`STACKLAYER COUNT SPACING WIDTH PITCH`
	`…`
	`<Mx> <nx> <spacingx> <widthx> <pitchx>`
	`…`
	`END STACKLAYER`
	Here:
	• `STACKLAYER`: Specifies the LEF name of the layer.
	• `COUNT`: Specifies the number of stripes on the layer. The default `COUNT` value is `1`.
	• `WIDTH`: Specifies the stripe width in microns. The default `WIDTH` value is the minimum width in LEF.
	• `SPACING`: Specifies the stripe spacing in microns. The default `SPACING` value is the minimum spacing in LEF.
	• `PITCH`: Specifies the set-to-set distance in a stripe. It indicates the range that one stripe set can cover. The default `PITCH` value is `MAX`, which means one set.
	`COUNT`, `WIDTH`, or `SPACING` can be omitted in the configuration file by specifying a hyphen (-). For example, `2` stripes on `M4` with minimum width and spacing can be described as:
	`M4 2 - -`
	If the entire line of a layer is omitted, it means that the layer will have one stripe with minimum width and spacing.
	Following is an example of a pattern defined in the stack layer configuration file:



The generation of each layer stapling depends only on its lower and upper stapling. The added PG can be shifted to find a DRC clean location. If a DRC clean location is not found, the added PG of this pin is removed and kept open. The length of the `Mx` layer depends on location of the `Mx-1` and `Mx+1` layers, as shown in the figure below. The number of wires created on the `Mx-1` and `Mx+1` layers needs to fulfill the defined count number criteria.

`-target_layer {layer1 layer2}`	Defines one or two destination power/ground stripes by specifying the target layer names. In case two target layers are specified, those should be adjacent. In such a case, the software connects to the closet wires of these two adjacent target layers with DRC clean. The directions of these two target layers are orthogonal. The configuration file is created based on the higher of these two layers. The lower target layer uses its own configuration file and the configuration files of the layers between the cell pin.
	For example, the `stack.cfg` configuration file for a stack is created based on the target layer `M9`. Its target layer `M8` uses the configuration files of the layers from `M3` to `M7`, as shown below:
	`add_target_pg -nets {VDD2} -target_layer {M9 M8} -pin_layer M2 -stack_layer_config stack.cfg`
	Following are the contents of the `stack.cfg` configuration file:
	`STACKLAYER COUNT SPACING WIDTH PITCH`
	`M3 2 0.12 0.04 0.36`
	`M4 2 0.12 0.04 0.9`
	`M5 - - 0.04 0.36`
	`M6 2 0.12 0.04 0.9`
	`M7 2 0.2 0.2 1.2`
	`M8 - 0.2 0.2 1.2`
	`END_STACK_LAYER_CONFIG`
	Following is the output of the above command:
	 
	As shown in the above output, since the first layer is not DRC clean, the software searches for the second closest target stripe to connect.
	Data_type: string, required

`-user_class subclass_string`	Indicates the name string of the user-defined attribute subcalss of wires to route. *Data_type*: string, optional

Examples

- The following command connects different cells and pins specifying the `-target_layer` as `M8` and `-pin_layer` as `M1`:
  ```
  set_db add_target_pg_share_resource false
  set_db add_target_pg_pins {cell1:VDD cell1:VSS cell2:VDD cell2:VSS}
  ```

```
add_target_pg -nets {VDD VSS} -target_layer M8 -pin_layer M1 -stack_layer_config_file  target_pg.config
```

Following are the `target_pg.config` file contents:

```
STACKLAYER COUNT  SPACING  WIDTH

M2           2            0.06              0.04

M3           2            0.12              -

M4           2            0.12              0.04

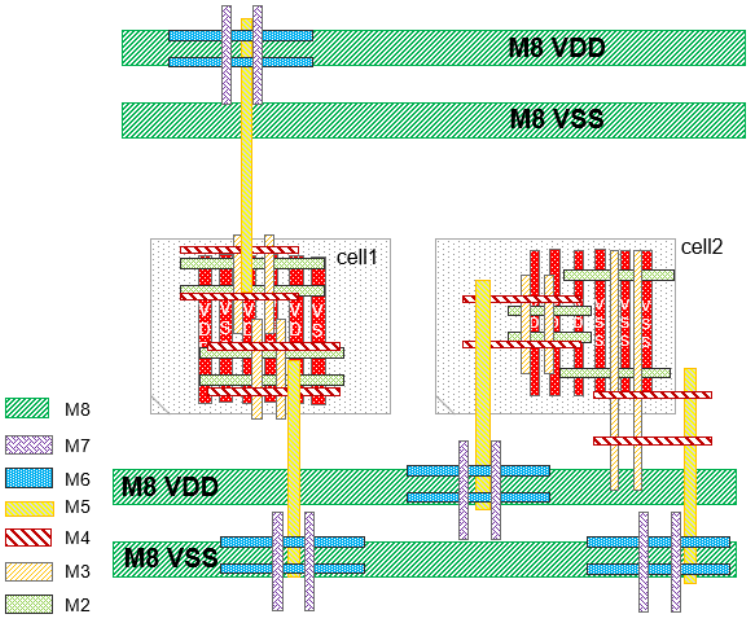
M5           -         -                  0.04

M6           2            0.12               0.04

M7           2             0.2               0.2

END STACKLAYER
```

Following is the output of this example:



- The following set of commands connect different cells and pins specifying the `-target_layer` as `M8` and `-pin_layer` as `M2`:

```
set_db add_target_pg_share_resource false
set_db add_target_pg_pins {cell1:VDD cell1:VSS cell2:VDD cell2:VSS}
add_target_pg -nets {VDD} -target_layer M8 -pin_layer M2 -stack_layer_config_file  target_pg.config
```

Following are the `target_pg.config` file contents:

```
STACKLAYER   COUNT  SPACING  WIDTH  PITCH

M3           3            0.12          -            -

M4           2            0.12          0.04         0.5

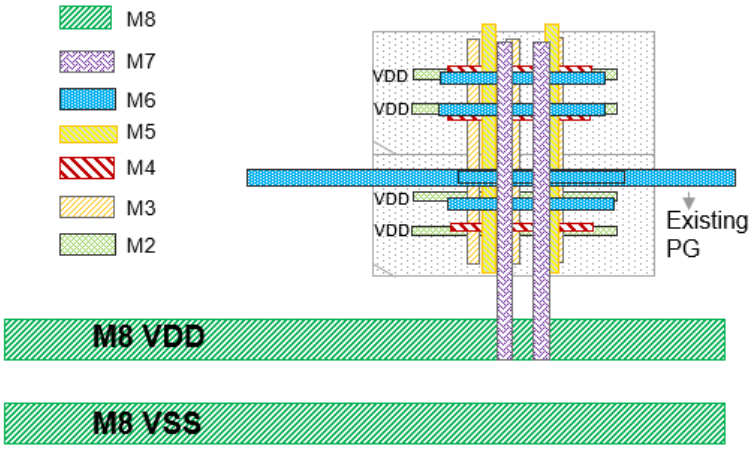
M5           2            0.12          0.04         -

M6           2            -             0.08         0.5

M7           2            0.2           0.2          -

END STACKLAYER
```

Following is the output of this example:



Related Information

`add_target_pg` Category Attributes

add_target_pg Category Attributes

The root attributes in the `add_target_pg` category are:

- add_target_pg_allow_weak_connect
- add_target_pg_max_extension_distance
- add_target_pg_pins
- add_target_pg_pins_group_distance
- add_target_pg_respect_routes
- add_target_pg_share_resource

To set the value of any root attribute, use the `set_db` command:

```
set_db attribute_name value
```

To get the current value of any root attribute, use the `get_db` command:

```
get_db attribute_name
```

To get the current value of all attributes in the `add_target_pg` category, use the following command:

```
get_db -category add_target_pg
```

add_target_pg_allow_weak_connect

`add_target_pg_respect_clock_routes {true | false}`

Default: `false`

Data_type: bool, read/write

When set to `false`, the number of wires created on every stack layer need to satisfy the defined count with DRC clean. Else, the PG pin is left open. When set to `true`, the count lesser than defined can also be accepted while creating the total count causing DRC.

add_target_pg_max_extension_distance

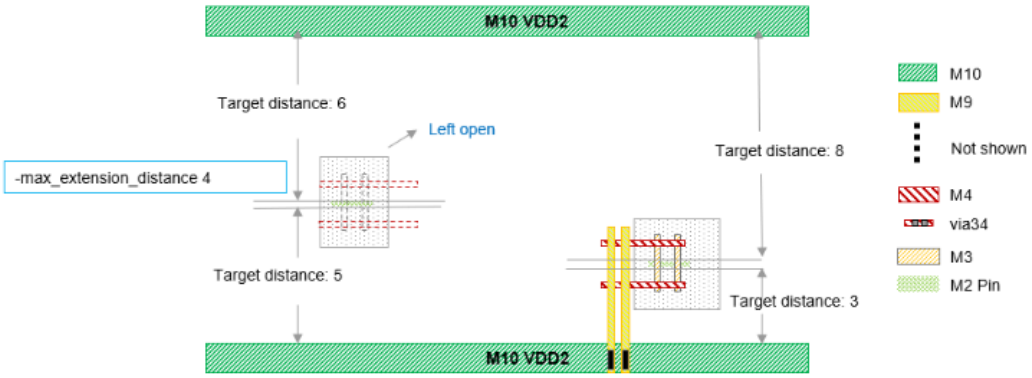
`add_target_pg_max_extension_distance value`

Default: `2147483647`

Data_type: double, read/write

Specifies the maximum distance that the target PG pin can extend to the target layer. If the distance between the target PG pin and the candidate target layer is lesser than or equal to the specified distance, the candidate target is treated as the real target.

If the target PG wire cannot be found within `add_target_pg_max_extension_distance`, the instance is left open. If this attribute is not used, the software connects the stripe layer to the closest target PG wire with DRC clean, by default. See the figure below for understanding:



add_target_pg_pins

`add_target_pg_pins {cell1:pin1 cell1:pin2 cell2:pin1 ...}`

Default: `""`

Data_type: string, read/write

Specifies the cell pins to be connected.

add_target_pg_pins_group_distance

`add_target_pg_pins_group_distance value`

Default: `auto`

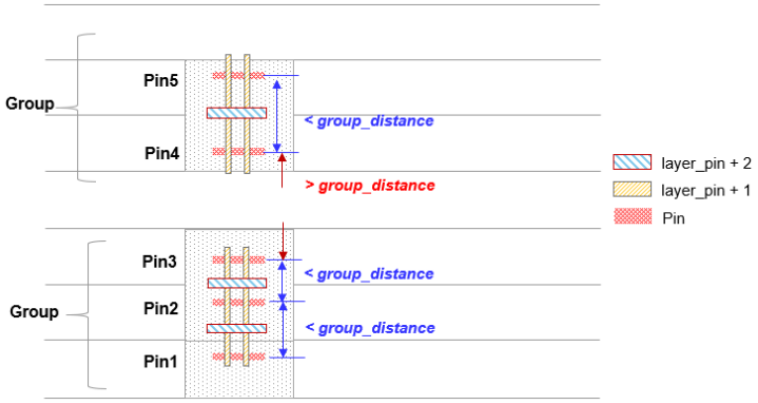
Data_type: string, read/write

Specifies the side-to-side distance of the pins to be treated as a group while connecting the target PG pins. The pins belonging to the same group share the same target PG stripes connection.

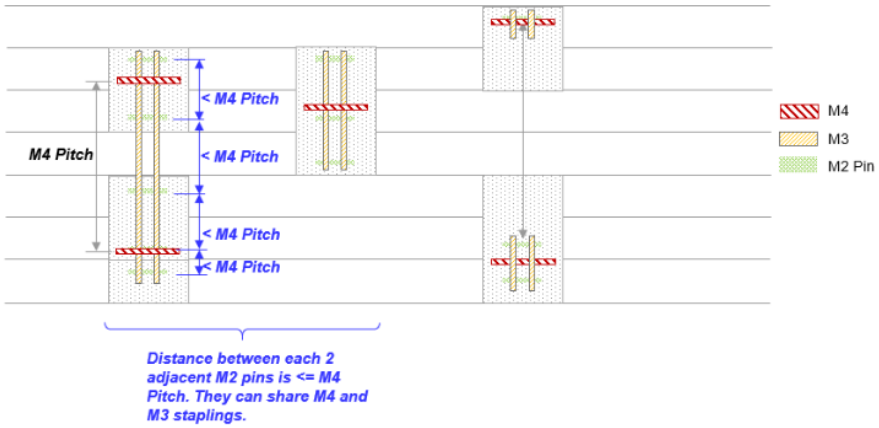
When this attribute is set to its default value, `auto`, the `add_target_pg` command uses the default behavior of target PG generation to group the pins. If this attribute is not specified, the software uses the pitch of metal layer above the pins layer with same direction (`layer_pin+2`) as the group distance.

If the pitch of metal layer above the pins layer with the same direction (`layer_pin+2`) is not defined, the pins will be connected separately.

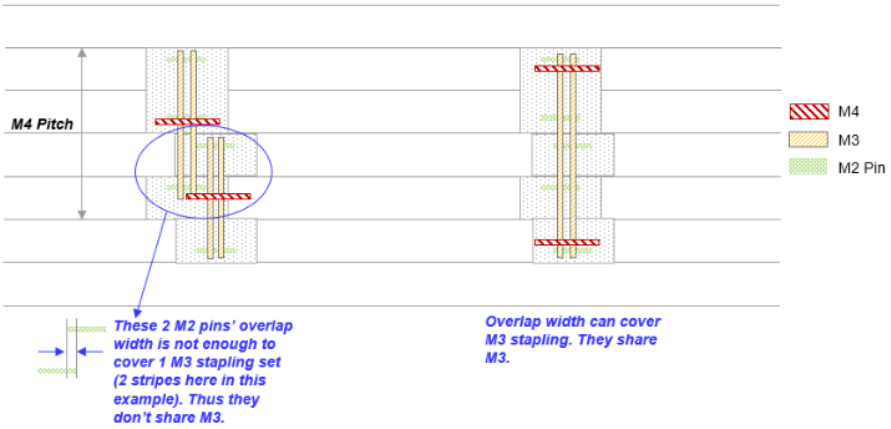
If the distance between two pins is lesser than `group_distance`, they will be treated as one group. See the figure below for understanding:



When `pins_group_distance` is not defined, the software uses the pitch of the metal layer above the pins layer with the same direction (`layer_pin+2`) as the group distance. See the figure below for understanding:



The following figure presents how the target PG pin's horizontal offset affects the stapling sharing:



add_target_pg_respect_routes

`add_target_pg_respect_routes {none | fixed | fixed_and_clock | all}`

Default: `fixed_and_clock`

Data_type: enum, read/write

Specifies the route types to respect for DRC check during target PG generation. This attribute always honor all covers and PGs, but does not honor the trail routes, Following are the attribute values:

- `none`: Specifies that no route types are respected.
- `fixed`: Respects fixed regular wires.
- `fixed_and_clock`: Respects fixed and routed clocks.
- `all`: Respects fixed and routed clocks along with the routed signals.

The following table presents the support of the `add_target_pg_respect_routes` attribute values with various statuses of net:

Target PG Honoring a Wire			add_target_pg_respect_routes Values			
Net Use	**Status**		**none**	**fixed**	**fixed_and_clock**	**all**
P/G			Yes	Yes	Yes	Yes
Non-P/G	**Cover**		Yes	Yes	Yes	Yes
	Fixed		No	Yes	Yes	Yes
	Routed	**Clock**	No	No	Yes	Yes
		Signal	No	No	No	Yes
	Unknown		No	No	No	No

add_target_pg_share_resource

`add_target_pg_share_resource {true | false}`

Default: `true`

Data_type: bool, read/write

When set to `true`, the newly added PGs of the intermediate stacked layer can connect to the existing PG wire of the same layers and are preferred to be shifted to the track of the existing PG to save the track resource. If there are DRCs between the added PG and the existing PG after shifting, the

added PG will follow the behavior explained below:

- If the added wire is on different net from the existing PG, it will be shifted again to avoid DRC.
- If the added wire is on the same net of the existing PG, it will be merged with the existing wire.

When set to `false`, the stacked layer cannot connect to the existing intermediate layers between the cell PG pin and target PG layer. Additionally, the newly added PG will be shifted if its original location causes DRC.

generate_special_via Category Attributes

The root attributes in the generate_special_via category are:

- generate_special_via_accuracy_effort
- generate_special_via_add_pin_to_pin_vias
- generate_special_via_align_merged_stack_via_metal
- generate_special_via_allow_via_expand
- generate_special_via_allow_wire_shape_change
- generate_special_via_area_only
- generate_special_via_check_signal_routes
- generate_special_via_create_double_row_cut_via
- generate_special_via_create_max_row_cut_via
- generate_special_via_cut_class_preference
- generate_special_via_disable_via_merge
- generate_special_via_enable_check_drc
- generate_special_via_extend_out_wire_end
- generate_special_via_full_cut_via_only
- generate_special_via_hookup_contact_max
- generate_special_via_hookup_contact_pg_track
- generate_special_via_hookup_fixed_grid
- generate_special_via_hookup_min_distance
- generate_special_via_hookup_rail_pair
- generate_special_via_hookup_via_distance
- generate_special_via_hookup_via_style
- generate_special_via_hookup_via_rule
- generate_special_via_hookup_virtual_trim_grid
- generate_special_via_ignore_design_boundary
- generate_special_via_ignore_drc
- generate_special_via_ignore_rule_enclosure
- generate_special_via_inherit_wire_status
- generate_special_via_keep_existing_via
- generate_special_via_keep_fixed_via
- generate_special_via_mincut_preference
- generate_special_via_opt_cross_via
- generate_special_via_opt_via_on_routing_track

- generate_special_via_parameterized_via_only
- generate_special_via_partial_overlap_threshold
- generate_special_via_preferred_vias_only
- generate_special_via_prefix
- generate_special_via_reference_boundary
- generate_special_via_respect_stdcell_geometry
- generate_special_via_respect_wire_ndr
- generate_special_via_rule_preference
- generate_special_via_set_via_expand_dir
- generate_special_via_snap_via_center_to_grid
- generate_special_via_split_long_via_global_grid
- generate_special_via_symmetrical_via_only
- generate_special_via_use_track_offset
- generate_special_via_use_trim_metal_enclosure

To set the value of any root attribute, use the `set_db` command:

```
set_db attribute_name value
```

To get the current value of any root attribute, use the `get_db` command:

```
get_db attribute_name
```

To get the current value of all attributes in the generate_special_via category, use the following command:

```
get_db -category generate_special_via
```

The following commands use the `generate_special_via` category attributes for various functionalities:

- `update_power_vias`
- `add_stripes`
- `add_rings`
- `add_target_pg`
- `add_reinforce_pg`
- `route_special`

generate_special_via_accuracy_effort

`generate_special_via_accuracy_effort {low | medium | high}`

Default: `high`

Data_type: enum, read/write

Controls the balance between accuracy and runtime during via generation.

- When set to `high`, viagen checks full DRC rules and creates DRC clean vias.
- When set to `medium`, viagen runs DRC check in the moderate mode.
- When set to `low`, viagen runs in the quick DRC check mode. With this value specified, some available vias may be missed, since the conservative rules are checked.

The vias generated through this parameter are DRC clean. This parameter generates vias with reduced runtime in the special scenarios where complex DRC rules are triggered. These complex rules include (but are not limited to) `same metal aligned cuts spacing`.

generate_special_via_add_pin_to_pin_vias

`generate_special_via_add_pin_to_pin_vias {true | false}`

Default: `false`

Data_type: bool, read/write

Inserts vias to connect pins between different cover macros.

generate_special_via_align_merged_stack_via_metal

`generate_special_via_align_merged_stack_via_metal {true | false}`

Default: `false`

Data_type: bool, read/write

Aligns merged stack vias.

generate_special_via_allow_via_expand

`generate_special_via_allow_via_expand {true | false}`

Default: `true`

Data_type: bool, read/write

Allows via to expand out of intersection area to meet LEF rules.

generate_special_via_allow_wire_shape_change

`generate_special_via_allow_wire_shape_change {true | false}`

Default: `true`

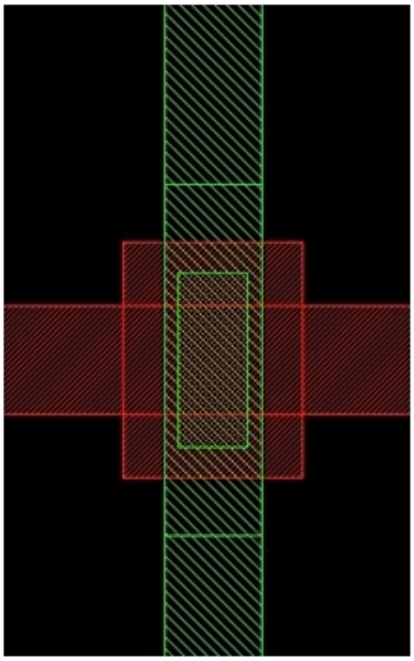
Data_type: bool, read/write

Allows the via to change the metal shape of existing wires.

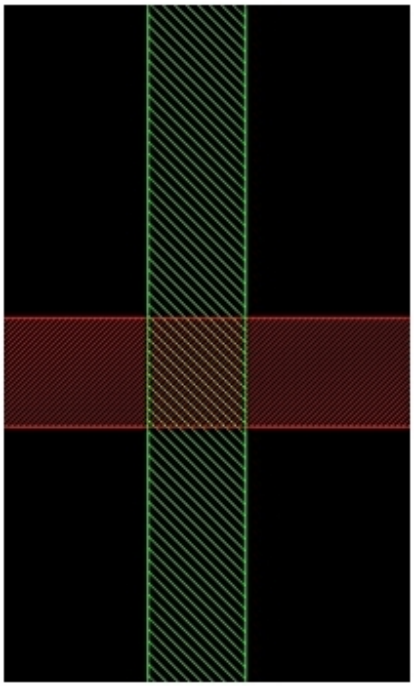
Note: For stack vias, the option is only valid for top and bottom metal layers.

Example:

- `generate_special_via_allow_wire_shape_change true`



- `generate_special_via_allow_wire_shape_change false`



generate_special_via_area_only

`generate_special_via_area_only {true | false}`

Default: `false`

Data_type: bool, read/write

Adds vias only within the specified areas.

generate_special_via_check_signal_routes

```
generate_special_via_check_signal_routes {0 | 1 | 2}
```

Default: `0`

Data_type: enum {0 1 2}, read/write

Controls whether the via generation software honors pre-existing signal routes.

If set to `1`, instructs the via generation software to read pre-existing signal routes and avoid creating DRC violations for the routes.

If set to 2, instructs the via generation software to read pre-existing signal routes and avoid creating DRC violations for the routes. And improve the runtime of VIA12 process.

Note: Using this setting can affect software performance significantly.

If set to `0`, instructs the via generation software to ignore any pre-existing signal routes.

Note: DRC violations can occur if this setting is used.

generate_special_via_create_double_row_cut_via

```
generate_special_via_create_double_row_cut_via {0 | 1 | 2}
```

Default: `0`

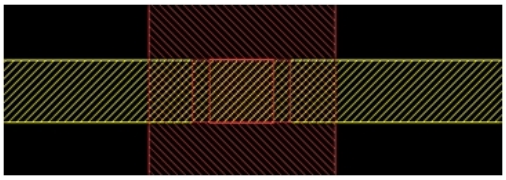
Data_type: enum {no_extra_cut add_extra_cut add_extra_cut_and_metal}, read/write

Try to expand vias to have one additional cut than original when set the option to 1 or 2. That is, expand original vias of `1xn` or `nx1`cuts to vias of `1x(n+1)` or `(n+1)x1` cuts.

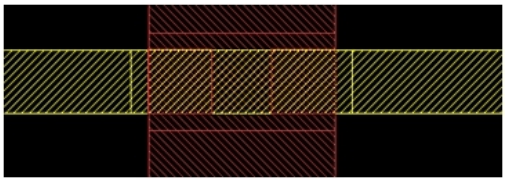
- 1: create vias with one additional cut where wires on both top and down layers are enough and need no shape change.
- 2: create vias with one additional cut where wires on one of the layers are enough.

Example:

- `generate_special_via_create_double_row_cut_via false`



- `generate_special_via_create_double_row_cut_via true`



generate_special_via_create_max_row_cut_via

```
generate_special_via_create_max_row_cut_via {true | false}
```

Default: `false`

Data_type: bool, read/write

Try to expand vias to have one additional row or column of cuts than original when the option is set to true. It works similar to `generate_special_via_create_double_row_cut_via` except that there is no restriction on one row or one column via.

generate_special_via_cut_class_preference

```
generate_special_via_cut_class_preference {default | {[square] [bar] [large]} | cutclass_name_list | file_name}
```

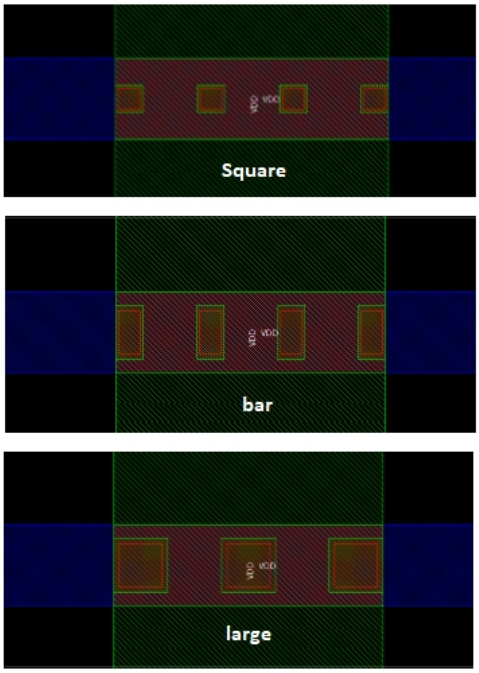
Default: `default`

Data_type: string, read/write

Specifies the preference for the via cut class while inserting special vias. You can specify one of the following values:

- `default`: Prefers to use the first viacell in tech LEF without drc violation or the first generate viarule to generate maximum cut area. For example, `cutWidth * cutHeight * cutNumber`.
- `square`: Prefers to use the SINGLE cut vias. In the LEF definition, the cut width and length are usually the same and the smallest in that layer. For example, 0.032 by 0.032.
- `bar`: Prefers to use the DOUBLE cut vias. In the LEF definition, the cut width is the usually the smallest. For example, 0.032 by 0.08. It mostly equals to 2 SINGLE cut.
- `large`: Perfers to use the LARGE cut vias. In the LEF definition, the neither the cut width not the length is the smallest. For example, 0.08 by 0.08, equals to 4 SINGLE cut.

The following figure shows the difference between the square, bar, and large values:



Note:

1. The attribute setting does not support incremental specification. The last `cutClass` preference setting is used.
2. When you specify `{[square] [bar] [large]}`, there should be 2 or 3 `cutClass` defined in the LEF file. If Innovus detects that there are more than 3 or less than 2 `cutClasses`, the setting is ignored.
3. In the specified file, different arguments can be separate by space or line break.

generate_special_via_disable_via_merge

```
generate_special_via_disable_via_merge {true | false}
```

Default:

- `false` in normal scenarios

- `true` for advance nodes (with designs of 12 nm or below)

Data_type: bool, read/write

Disable via merging in the layers next to the followpin layer within the range of two when set the option to true and there is followpin layer in the design. Triming will be the only way to solve drc issues.

For example, if the followpin layer is in metal1, disable via merging in metal1/metal2/metal3; if the followpin layer is in metal2, disable via merging in metal1/metal2/metal3/metal4.

This function helps if the merging of via consums additional wire track unnecessarily.

generate_special_via_enable_check_drc

```
generate_special_via_enable_check_drc {true | false}
```

Default: `false`

Data_type: bool, read/write

Enables or disables `check_drc` called by the via generation software.

- When set to `true`, the via generation software calls `check_drc` to detect any `min_step` violations, then calls the `fix_via -min_step` command to fix them. This may impact the run-time performance.
- When set to `false`, the via generation software will not call `check_drc`. This may leave `min_step` violations in the design.

Note: When both the `generate_special_via_enable_check_drc` and `generate_special_via_opt_cross_via` attributes are set to `true`, and the process node is above `10N`, DRC violations can be reported.

generate_special_via_extend_out_wire_end

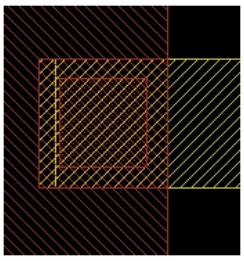
```
generate_special_via_extend_out_wire_end {true | false}
```

Default: `true`

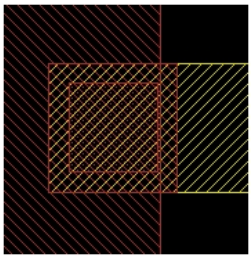
Data_type: bool, read/write

Allows via extending outside of existing wire ends at T-type of intersection. When set to false, via extending outside of the existing wire end are not allowed and may shift inside.

- `generate_special_via_extend_out_wire_end true`



- `generate_special_via_extend_out_wire_end false`



generate_special_via_full_cut_via_only

```
generate_special_via_full_cut_via_only {true | false}
```

Default: `false`

Data_type: bool, read/write

Allows the user to create only full cut vias.

Example
The following via is generated because of the second left cut will have drc violations when the option is set to false.

- `generate_special_via_full_cut_via_only false`

When set the option to true, only full cut vias will be generated:

- `generate_special_via_full_cut_via_only true`

generate_special_via_hookup_contact_max

```
generate_special_via_hookup_contact_max {distance1_max distance2_max}
```

Default: `{0, 0}`

Data_type: string, read/write

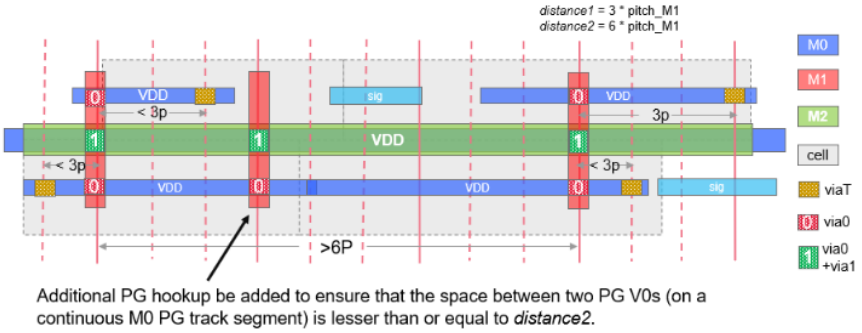
Sets the maximum distance to create hookup via from contact, and the maximum distance between two hookup via on a continuous `M0` PG pin. Following are the attribute values:

- `distance1_max`: Specifies the distance required by every viaT for at least one hookup via drop. For deciding the location of hookup via for `distance1_max`, every viaT needs at least one hookup via drop within `distance1`.
- `distance2_max`: Specifies the maximum distance between any two PG hookup vias on a continuous `M0` PG pin. An additional hookup via must be dropped from the previous hookup via on a continuous `M0` PG segment with `distance2_max`. Also, an additional hookup via must be dropped within the maximum `distance2` from the endpoint of a continuous `M0` PG pin.

Default: `0 0`

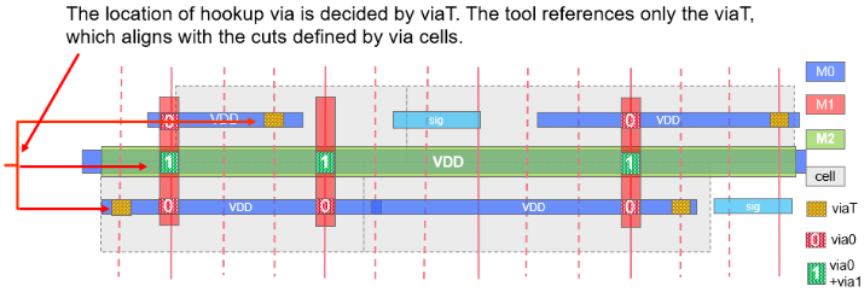
The following scenarios depict the usage of this attribute:

- Deciding the location of hookup via by viaT:

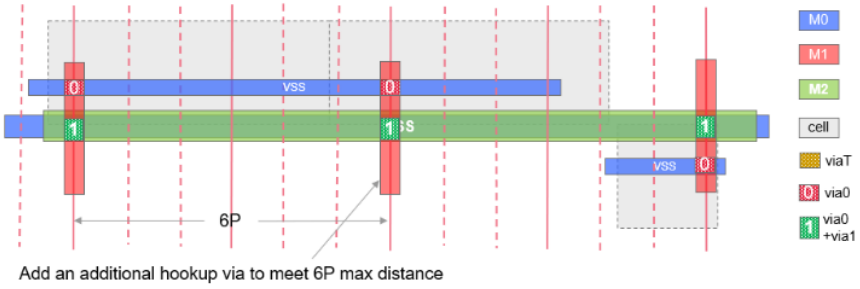


Additional PG hookup be added to ensure that the space between two PG V0s (on a continuous M0 PG track segment) is lesser than or equal to *distance2*.

- Referencing of the location of viaT by the tool to add hookup vias:



- At least one hookup via needs to be available for each continuous `M0` PG segment, including PG segment at both sides of the row edge and the PG pin. The continuous `M0` PG segment without viaT also needs to follow the decision rule of `distance2`:



- Four individual via cells are defined in the LEF file and VIAMAP. Of these, one is 3-cut, two are 2-cut, and the remaining one is 1-cut. The further breakup of these via cells is mentioned below:
  - 3-cut via cells: Include three `v0` cuts, three `m0` metals, and one `m1` metals
  - 2-cut via cells: Include two `v0` cuts, two `m0` metals, and one `m1` metals
  - 1-cut via cell: Include one `v0` cuts, one `m0` metals, and one `m1` metals

generate_special_via_hookup_contact_pg_track

```
generate_special_via_hookup_contact_pg_track {start_value pitch_value}
```

Default: `{0, 0}`

Data_type: string, read/write

Sets start point and the pitch of PG track used by hookup contact. Following are the attribute values:

- `start_value`: Specifies the location of the first PG track.
- `pitch_value`: Specifies the distance (steps) between two adjacent PG tracks. It needs to be an integer multiple of the `m1` pitch.

Note: The PG track needs to be aligned with the `m1` track.

generate_special_via_hookup_fixed_grid

```
generate_special_via_hookup_fixed_grid {true | false}
```

Default: `false`

Data_type: bool, read/write

When set to `false`, the added hookup via can be shifted to avoid DRC. When set as `true`, the hookup via can only be added at the fixed grid.

The fixed grid works with `generate_special_via_hookup_via_style`. For the `loose` and `moderate` styles of `generate_special_via_hookup_via_style`, the hookup via will be removed, if it has DRC while getting created. For the `compact` style of `generate_special_via_hookup_via_style`, the hookup via will be kept, whether or not it has DRC.

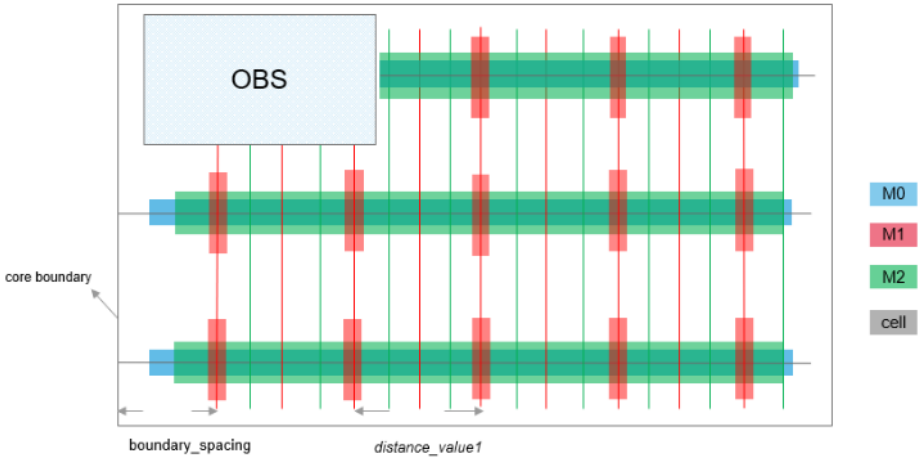
The fixed grid is controlled by the following attribute:

`set_db generate_special_via_hookup_via_distance {distance_value1 [distance_value2 [boundary_spacing]]}`

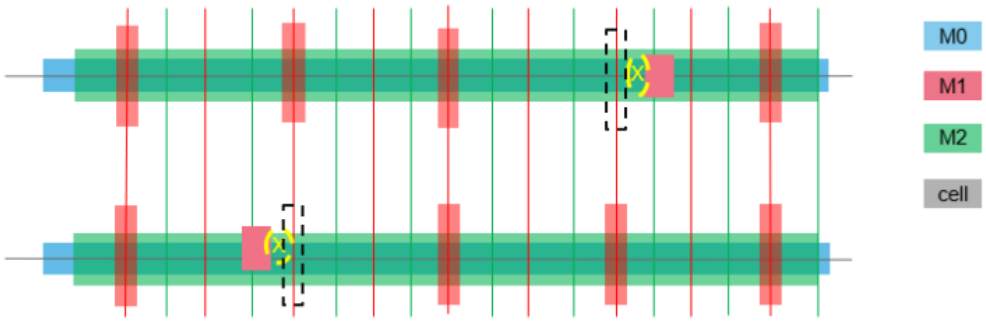
Here:

- *distance_value1* is the pitch of grid.
- *boundary_spacing* is the distance between the first grid and core boundary.
- The fixed grid does not get influenced by the shape of the design or macros.
- *distance_value2* needs to be set same as *distance_value1*.

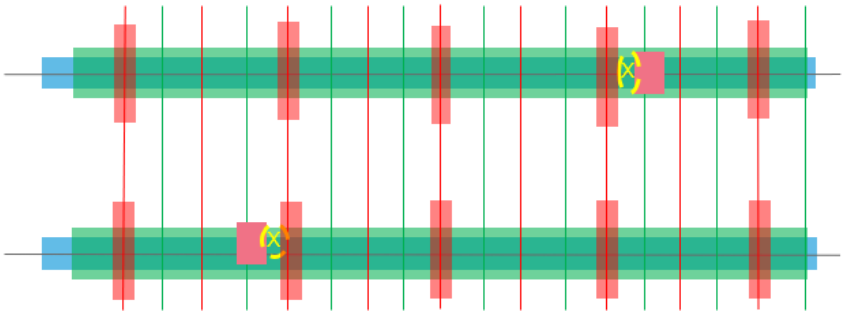
Below is an example of formation fixed grid using *boundary_spacing* and *distance_value1*. Note that all the vias of different lines need to be aligned.



Below is an example of DRC control for the `loose` and `moderate` types of hookup via styles. Here, the hookup vias are removed, if DRC exists.



Below is an example of DRC control for the `compact` type of hookup via style. Here, the hookup via will be kept, irrespective of the existence of DRC.



generate_special_via_hookup_min_distance

`generate_special_via_hookup_min_distance` *`string`*

Default: `min`

Data_type: string, read/write

Specifies the minimum distance between two PG vias. This attribute works with the distance between:

1. Two newly added PG vias
2. Existing PG vias and newly added PG vias on the followpin

generate_special_via_hookup_rail_pair

`generate_special_via_hookup_rail_pair {true | false}`

Default: `false`

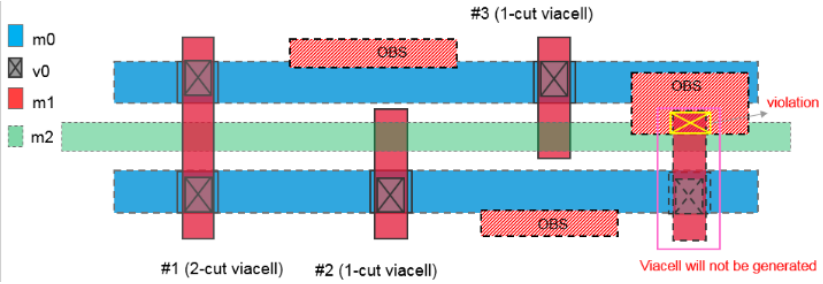
Data_type: bool, read/write

Adds hookup via over the pairing `M0` followpins that are distributed on both sides of a row with specially defined viacells at LEF file. Note that while using this attribute, the PG pins need to be on track, and the width of the `M0` followpin needs to be uniform.

This attribute works only with setting the `loose` value of `generate_special_via_hookup_via_style`. Three viacells are defined in the LEF file. One of those is 2-cut, and the remaining two are 1-cut. Refer to their details below:

- 2-cut viacell = Two `v0` cuts + Two `m0` metals + One `m1` metals
- 1-cut viacell = One `v0` cuts + One `m0` metals + One m1 metals

For the 1-cut viacell, `m1` needs to overlap completely with `M2`. If this is not possible, the viacell will not be generated. Refer to the example below:



generate_special_via_hookup_via_distance

`generate_special_via_hookup_via_distance {`*`distance_value1`* `[`*`distance_value2`* `[`*`boundary_spacing`*`]]}`

Default: `{4pitch 4pitch 0}`

Data_type: string

Enables control to put the first pg via for each followpin. You can use this attribute in the following ways:

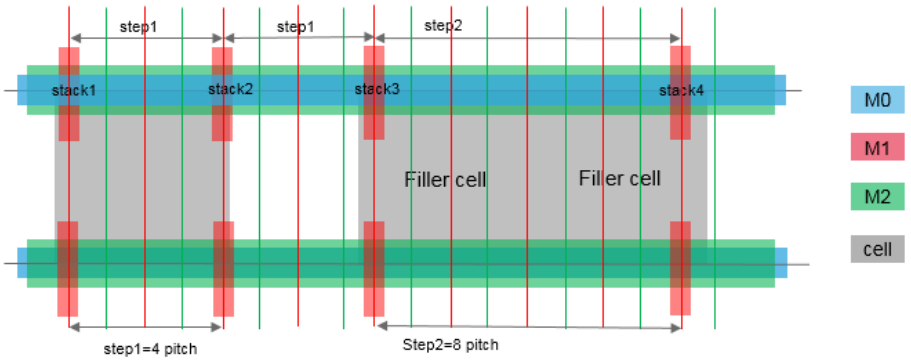
```
set_db generate_special_via_hookup_via_distance {distance_value1}
```

```
set_db generate_special_via_hookup_via_distance {distance_value1 distance_value2}
```

```
set_db generate_special_via_hookup_via_distance {distance_value1 distance_value2 boundary_spacing}
```

Following are the possible attribute values:

- *`distance_value1`*: Specifies the distance (step) between via stacks and is rounded down to `M1 pitch*n`. Default value is `M1 pitch*4`.
- *`distance_value2`*: Specifies the distance between via stacks on filler cell. This value can work with all hookup styles. Default value is `M1 pitch*4`. It indicates the via distance when two neighboring via stacks are over filler cells. Refer to the figure below:



- *boundary_spacing*: Specifies the distance between the first via and core boundary. It also refers the minimum distance between the last via and core boundary.

generate_special_via_hookup_via_style

`generate_special_via_hookup_via_style {none loose compact moderate}`

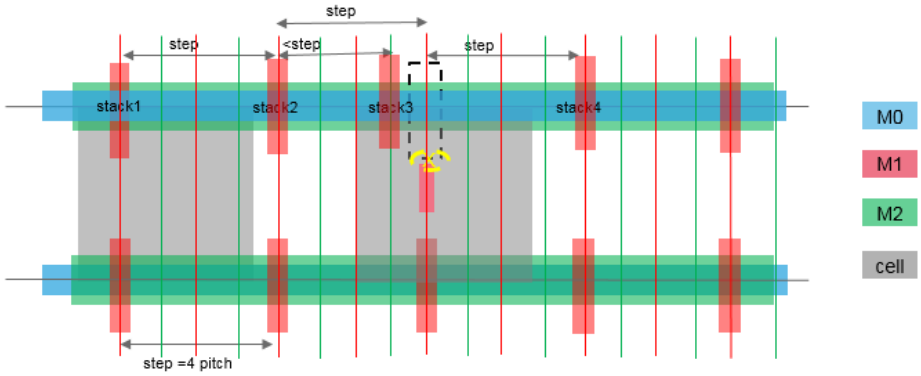
Default: `none`

Data_type: enum {none loose compact moderate}, read/write

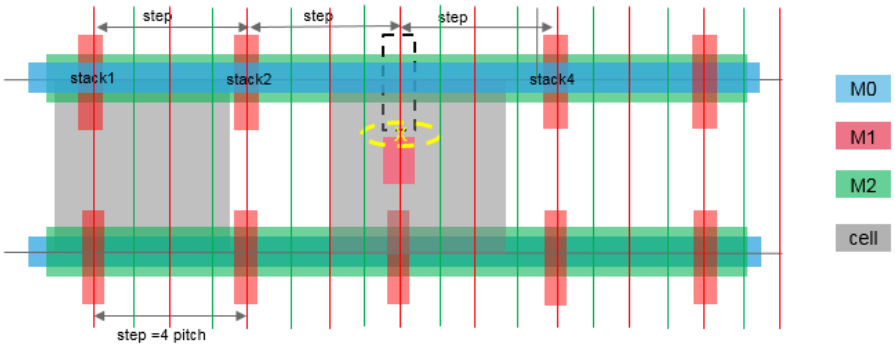
Sets hookup via style. The hookup via works between two parallel metal layers (like `M0` and `M2`, or `M1` and `M3`). Assuming it for `M0` and `M2`, viagen adds stack vias between `M0` and `M2` according to the hookup via settings.

The associated functionality of various options is provided below:

- `none`: No via hookup style is set. This is the default value.
- `loose`: The tool inserts vias between paralleled wires with the specified step in `generate_special_via_hookup_via_distance`. The center of the left-most via needs to be on the track, which is inside the intersection. The via also needs to be inside the intersection. If a via stack (like `stack3`) cannot be dropped at a certain step position (dot line area in the figure below) because of the DRC violation, it is moved to a closer or farther position with the previous stack (`stack2`) to ensure DRC clean. Additionally, the next via (`stack4`) needs to keep the step value with the original position of moved stack (dot line area in the figure below).

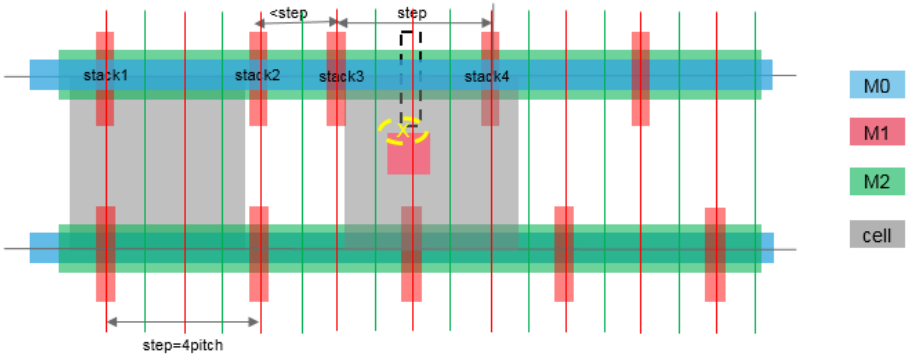


If DRC clean does not occurs, the via stack is removed, refer to the figure below:

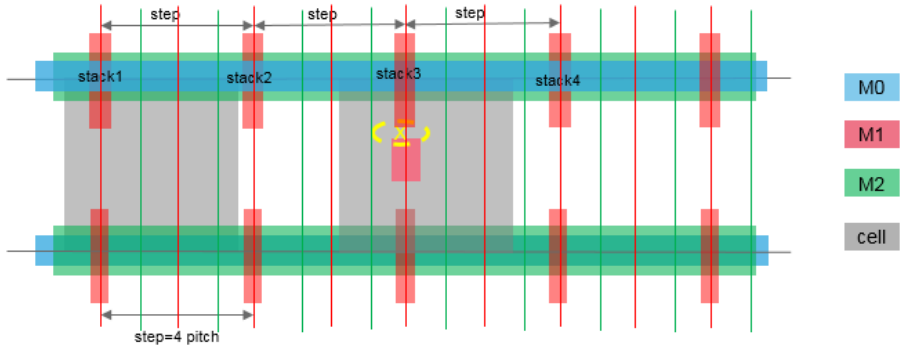


Note: The hookup style is `loose` by default, if the via map table does not exist.

- `compact`: If a via stack (like `stack3`) cannot be dropped at a certain step position (dot line area in the figure below) because of DRC violation, it is moved to a closer position with the previous stack (`stack2`) to ensure DRC clean. And the next via (`stack4`) needs to keep the step value with the said via (`stack3`). Refer to the figure below:

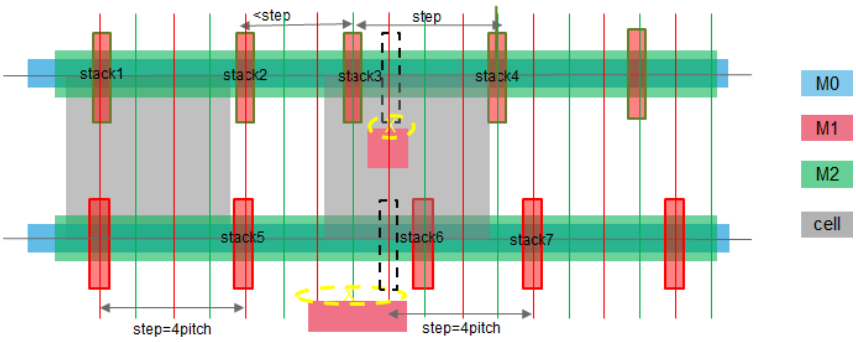


If DRC clean does not occurs by moving the stack, insert the via stack at the original position with DRC violation marker. Refer to the figure below:

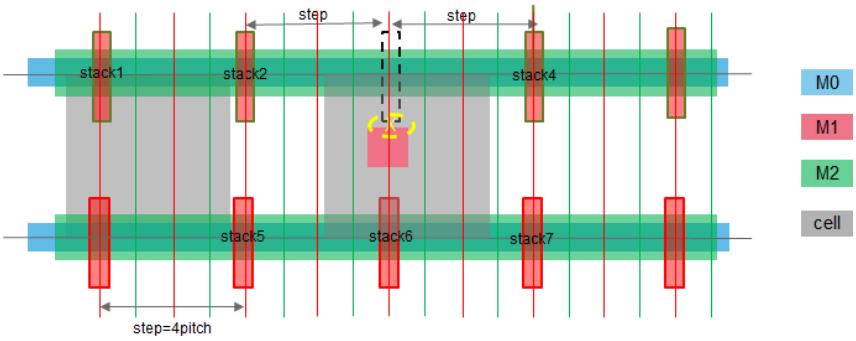


Note: The hookup style is `compact` by default, if the via map table exists.

- `moderate`: If a via stack (like `stack3`) cannot be dropped at a certain step position (dotted line area in the figure below) due to the DRC violation, it is moved to a closer or farther position with the previous stack (`stack2`) to ensure DRC clean. If the via stack (`stack3`) is dropped to a closer position with the previous stack (`stack2`), the next via (`stack4`) needs to keep the step value with the said stack (`stack3`). If the via stack (`stack6`) is dropped to a farther position with the previous stack (`stack5`), the next via (`stack7`) needs to keep the step value with the original position of the moved stack.



If DRC clean is not occurred, `viaGen` checks the next tracks followed to insert the via stack at the nearest DRC clean track.



generate_special_via_hookup_via_rule

`generate_special_via_hookup_via_rule` *string*

Default: `""`

Data_type: string, read/write

Specifies the preferred viarule for the designated cell area during the hookup via generation. This attribute can also work with the `generate_special_via_preferred_vias_only` attribute to use different types of viarule orders for filler cell to be DRC clean.

generate_special_via_hookup_virtual_trim_grid

`generate_special_via_hookup_virtual_trim_grid {-layer` *layerNumber* `-mask1 {-pitch` *value* `-core_offset` *value*`} -mask1 {-pitch value ...} -mask2 {-pitch value ...} -mask2 {-pitch value ...}}`

Default: `""`

Data_type: string, read/write

Creates a virtual trim grid for PG hookup via generation. The created virtual trim grid should align with the via metal of the newly created hookup via. When this attribute is specified, the existing trim grid is ignored.

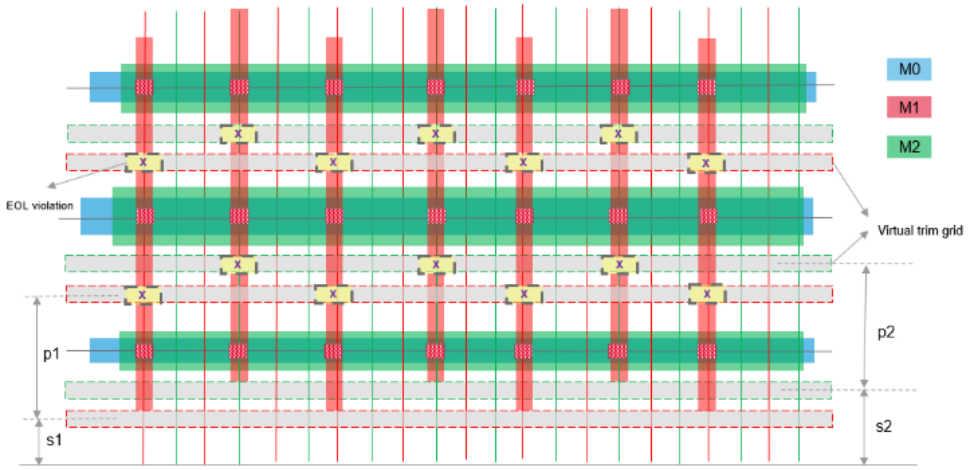
To honor the existing trim grid, either remove this attribute or define the existing trim grid within this parameter. `-core_offset` is calculated from the bottom core boundary to the center of the first grid.

Note: The width value for this attribute is taken from the LEF file.

Following is an example of creating a virtual trim grid for PG hookup via generation:

```
generate_special_via_hookup_virtual_trim_grid -hookup_virtual_trim_grid {-layer M1 -mask1 {-pitch p1 -core_offset s1 } -mask2 { -pitch p2 -core_offset s2 }}
```

Following is the output of the above command:



generate_special_via_ignore_design_boundary

`generate_special_via_ignore_design_boundary {true | false}`

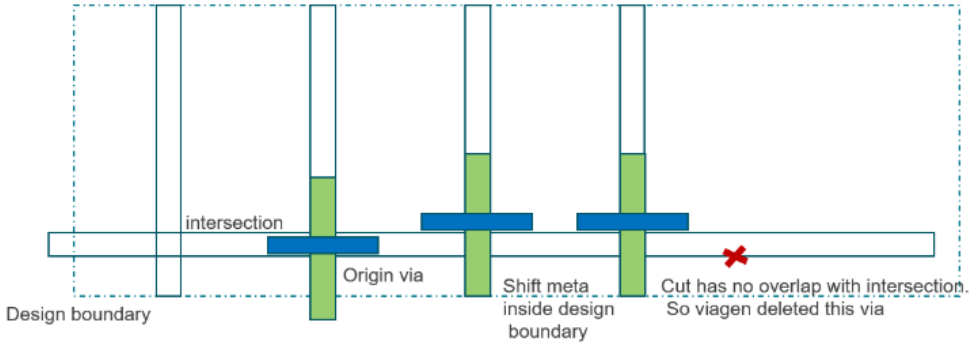
Default: `false`

Data_type: bool, read/write

Specifies whether a via can be extended out of the design boundary (floorplan area).

- When set to `true`, viagen generates vias and ignores the design boundary.
- When set to `false`, viagen tries to shift vias first inside the boundary area. If it does not overlap with the intersection point after shifting, viagen deletes it. If it overlaps with the intersection point after shifting with DRC clean, viagen keeps it.

See the figure below for understanding the usage of this attribute:



generate_special_via_ignore_drc

`generate_special_via_ignore_drc {true | false}`

Default: `false`

Data_type: bool, read/write

Setting this option ignores DRC but honors user-defined values when creating vias. This speeds up via creation during power-planning but normally causes DRC violations.

It should only be used during early prototyping to experiment with different power-planning choices that will be updated later to be DRC clean.

Note: This attribute ignores only the DRC violations from third party objects, and the software will still check the violation inside via.

generate_special_via_ignore_rule_enclosure

`generate_special_via_ignore_rule_enclosure {true | false}`

Default: `true`

Data_type: bool, read/write

When set to `true`, the software ignores enclosure value in VIARULE GENERATE and uses the optimized enclosure from cut layer rules, to get the maximum number of cuts, minimum via metal shape. When set to `false`, the enclosure value in VIARULE GENERATE must be satisfied. When this VIARULE fails DRC check, the software takes next VIARULE or viacell from LEF file.

generate_special_via_inherit_wire_status

`generate_special_via_inherit_wire_status {true | false}`

Default: `false`

Data_type: bool, read/write

Setting this option true to create via and set the status of the via to that of the target wire. When set the option to false, via statuse will be set to ROUTED.

For `update_power_vias`, target wire is the bottom wire. For other commands calling VIAGEN engine, whether bottom or top wire is the target wire depends on order of wire read-in. So if this option is designed to use, `update_power_vias` is the recommended command.

Note: `update_power_vias` does not delete or modify vias having status as FIXED/COVER.

generate_special_via_keep_existing_via

`generate_special_via_keep_existing_via {0 | 1 | 2}`

Default: `0`

Data_type: enum {remove keep_with_new_via keep_with_no_new_via}, read/write

If a via exists in an intersection and does not cover the whole intersection, a new via is tried to be dropped in the same intersection. Following are the attribute values:

- `0`: Prevents keeping an existing via, while dropping a new via.
- `1`: Keeps an existing via, while dropping a new via.
- `2`: Keeps an existing via and does not drop a new via at the same intersection, even when the existing via does not cover the whole intersection area.

generate_special_via_keep_fixed_via

`generate_special_via_keep_fixed_via {true | false}`

Default: `true`

Data_type: bool, read/write

Controls whether to preserve the fixed vias or not when `viaGen` is called for via generation.

generate_special_via_mincut_preference

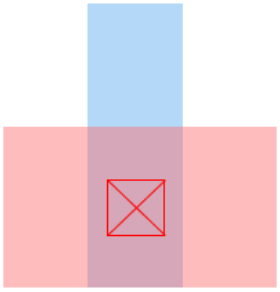
`generate_special_via_mincut_preference` *`int_value`*

Default: `1`

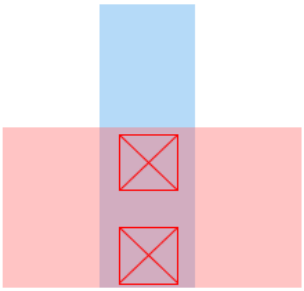
Data_type: int, read/write

Specifies the minimum number of cuts during via generation at an intersection. For example:

- If this attribute is set to `1`, viagen generates at least one cut at an intersection, as shown below:



- If this attribute is set to `2`, viagen generates at least two cuts at an intersection, as shown below:



generate_special_via_opt_cross_via

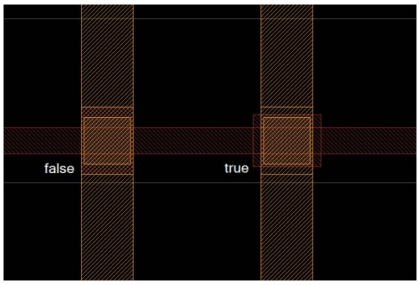
`generate_special_via_opt_cross_via {true | false}`

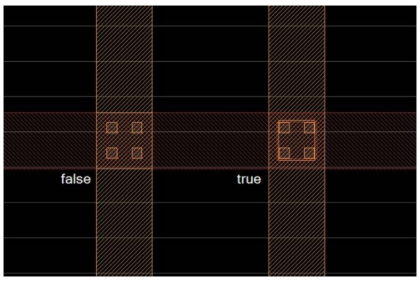
Default:

- `true` for advanced node (10 NM and 7 NM) designs.
- `false` for other designs.

Data_type: bool, read/write

Generate vias with the minimum enclosure value on each layer. The different result with false or true are shown below:





generate_special_via_opt_via_on_routing_track

```
generate_special_via_opt_via_on_routing_track {true | false}
```

Default: false

Data_type: bool, read/write

Free up routing tracks by shift or modify the vias minimally in intermediate layers and not applied to layers where the wires or pins exist. This option only supports generated vias and not predefined vias.

Note: Free up routing tracks against forbidden spacing rule for cross via and save routing tracks against endofline spacing rule for non cross via. `generate_special_via generate_special_via_opt_cross_via 1` will try to generate cross via.

generate_special_via_parameterized_via_only

```
generate_special_via_parameterized_via_only {auto | true | false}
```

Default: `auto`

Data_type: enum {auto true false}, read/write

When the value of this attribute is set to:

- `true`, ViaGen generates parameterized vias only, which means the DEF syntax of generated vias is with VIARULE. It helps in OpenAcceess designs.
- `false`, ViaGen generates both parameterized and fixed vias.
- `auto`, it is `true` for OpenAccess design, `false` for non-OpenAccess design.

generate_special_via_partial_overlap_threshold

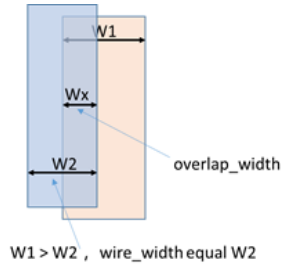
```
generate_special_via_partial_overlap_threshold float
```

Default: `0.0`

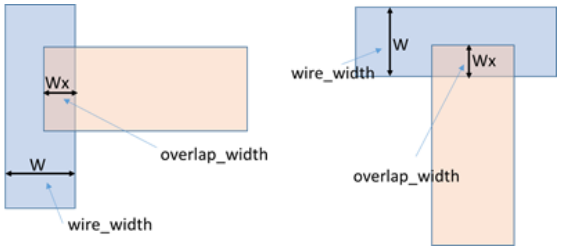
Data_type: double, read/write

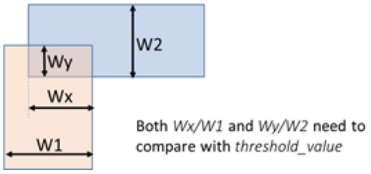
Specifies ratio threshold of via creation on partial overlap wires. The parameter value may range from `0` to `1`. The parameter value `1` specifies that no via is dropped on partial overlap intersection. The tool drops via only when `overlap_width`/`wire_width` is greater than or equal to the `threshold_value`.

The following figure depicts parallel overlapping wires:



The following figures depict vertical cross wires:





generate_special_via_preferred_vias_only

`generate_special_via_preferred_vias_only {use_lef | open | keep}`

Default: `use_lef`

Data_type: enum {use_lef open keep}, read/write

Generates only the vias set in `generate_special_via_rule_preference`. This attribute must be used with `generate_special_via_rule_preference`.

When the value of this parameter is set to:

- `use_lef`: viarule or via cell specified in `generate_special_via_rule_preference` is firstly considered. If none of them fits, viagen chooses the first DRC clean predefined viarule in the LEF file or auto viarule that can bring maximum cut area.cNote that:
  - Cut area = single cut area * cut number
  - Auto viarule includes generate viarule and virtual viarule. Virtual viarule means viarule generated by viagen based on the cut layer rule.
- `open`: Only predefined viarule or via cell specified in `generate_special_via_rule_preference` is considered. The first one with DRC clean is kept. If all of them have DRC issues, the whole stack is left open.
- `keep`: Only predefined viarule or via cell specified in `generate_special_via_rule_preference` is considered. The first one with DRC clean is kept. If all of them have DRC issues, keep the first one which is DRC clean with surrounding. If all of them have DRC issues with surrounding, the whole stack is left open.

For more information on predefined viarule and generate viarule, see the *Via Rule* and *Generate Via Rule* sections respectively of the LEF Syntax chapter of the *LEF/DEF Language Reference* document.

generate_special_via_prefix

`generate_special_via_prefix` *`viaNamePrefix`*

Default: `none`

Data_type: string, read/write

Sets the specified prefix name as the prefix for all generated vias created by Innovus. This helps user to select and edit the vias that need further modification.

generate_special_via_reference_boundary

`generate_special_via_reference_boundary {design | core}`

Default: `design`

Data_type: enum {design core}, read/write

Specifies the offset reference when specifying offset values in the `generate_special_via_split_long_via_global_grid` attribute. Following are the attribute values:

- `design`: ViaGen uses design boundary as reference.
- `core`: ViaGen uses core boundary as reference.

generate_special_via_respect_stdcell_geometry

`generate_special_via_respect_stdcell_geometry {true | false}`

Default: `false`

Data_type: bool, read/write

Respects geometry inside standard cell patterns.

generate_special_via_respect_wire_ndr

```
generate_special_via_respect_wire_ndr {true | false}
```

Default: `false`

Data_type: bool, read/write

Controls spacing between hookup via and signal routes. This allows viagen to support all NDR rules.

- When set to `false`, viagen honors a set of NDR rules, which include PG wire NDR, PG via NDR, Pin wire NDR, pin via NDR, and signal via NDR.
- When set to `true`, the signal wire NDR is also honored, including those listed above with the `false` value selection of this attribute. This means that setting this attribute to `true` honors all the NDR rules.

generate_special_via_rule_preference

```
generate_special_via_rule_preference {default | predefined | generated | <list of via rule/cell names> | <file name>}
```

Default: `default`

Data_type: string, read/write

Sets the preferred via-rule usage. Use this parameter before running the `fcroute`, `sroute`, or power planning commands.

- default
  Uses the first via rule defined in the LEF technology file as the preferred usage.
- `predefined`
  Uses the predefined via rule defined in the LEF technology file as the preferred usage.
- `generated`
  Uses the generated via rule defined in the LEF technology file as the preferred usage.
- `list of via rule/cell names`
  Uses the list of via rules/cells defined in the LEF technology file as the preferred usage.
- `file name`
  Uses via rules/cells which are contained in the specific file.

This attribute works in the incremental mode. See the example below:

```
set_db generate_special_via_rule_preference {V1 V2}
set_db generate_special_via_rule_preference {V3}
V1 V2 V3
```

generate_special_via_set_via_expand_dir

```
generate_special_via_set_via_expand_dir {auto | horizontal | vertical}
```

Default: `auto`

Data_type: enum {auto horizontal vertical}, read/write

Sets via expansion directions.

generate_special_via_snap_via_center_to_grid

```
generate_special_via_snap_via_center_to_grid { layer1 {none | grid | half_grid | either } layer2 {none | grid | half_grid | either } … }
```

Default: `none`

Data_type: string, read/write

Specifies the via center alignment style per routing layer.

`none`
Keeps previous behavior, only snaps to wire center line.

`grid`
Snaps via to the full grid layer track.

`half_grid`
Ignores the full grid. Only snaps to the `half_grid` layer track.

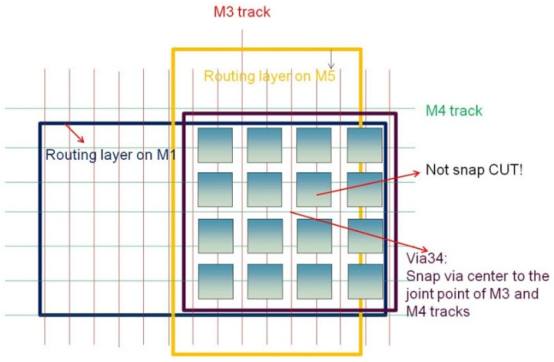
`either`
Can snap to both full and half grid layer track.

Note: Metal1 and Metal2 cannot be defined in option `snap_via_center_to_grid`. Innovus will not shift the via even when Metal1 and Metal2 are defined in the option.

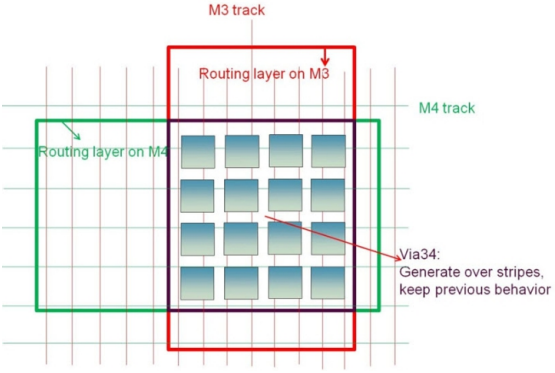
- V12 is not shifted in both vertical and horizontal directions
- V23 is not shifted in vertical direction

Examples

- **Snap the vias to each adjacent up and down layers, except Metal1 and Metal2**
  - NO stripes on up and down layers
  - Snap the center of via (the whole via, not CUT) to the closest track intersection

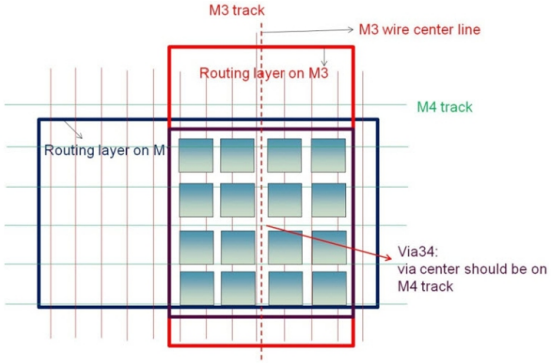


- **Stripes on both up and down layers**
  - Generate via on the stripes intersection area (the same as previous behavior, NO change)

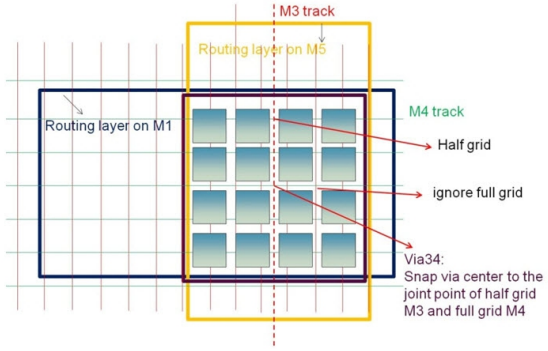


- **Stripe on adjacent layer metal3 and NO stripe on the other neighboring routing layer**
  - Find the joint point between stripe center and the closest track

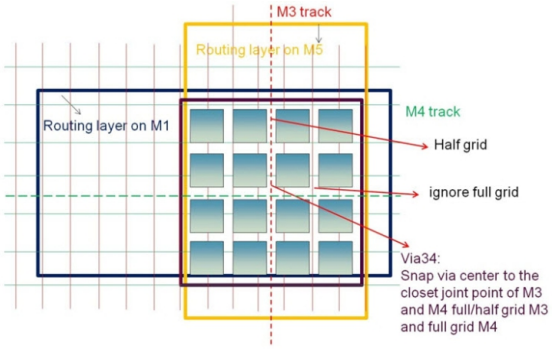
- Snap the center of via (the whole via, not CUT) to this joint point



- **Snap via to the full grid of metal4) and half grid of metal3**
  - `generate_special_via -snap_via_to_grid {metal3 half_grid metal4 grid}`
  - Snap via to metal3 half grid and metal4 full grid



- **Snap via to full or half grid of metal3 and metal4**
  - `generate_special_via -snap_via_to_grid {metal3 either metal4 either}`
  - Tool could snap to closest full or half grid automatically



generate_special_via_split_long_via_global_grid

`generate_special_via_split_long_via_global_grid { x_offset x_pitch x_length_threshold x_length_multiplier y_offset y_pitch y_length_threshold y_length_multiplier }`

Default: `default`

Data_type: string, read/write

Provides options to control global grids so that vias on the grids are in line on x and/or y directions.

- `x_offset, y_offset` : specifies in microns the offset from design boundry
- `x_pitch, y_pitch` : specifies in microns the centre to centre distance in split vias
- `x_length_threshold, y_length_threshold` : specifies vias longer or equal to threshold are being split
- `x_length_multiplier, y_length_multiplier` : specifies unit via length multiplier

generate_special_via_symmetrical_via_only

```
generate_special_via_symmetrical_via_only {true | false}
```

Default: `false`

Data_type: enum {auto true false}, read/write

By default the metal-width of a generated-via matches the width of the intersection of the metal above and below the via. If the width is an odd number of manufacturing grids (mgrids) wide, the resulting via cannot be represented by a parameterized via (LEF/DEF VIA with VIARULE and parameters or OpenAccess standardVia) because they require a symmetrical enclosure values on both sides of the cuts. In that case, a fixed-name via (LEF/DEF VIA with RECT statements or OpenAccess customVia) must be created to have the same width as the intersection area.

If this option is true, the via metal is enlarged one mgrid in the odd-mgrid directions order to make the via symmetrical so it can be represented as a parameterized via. This does not cause any DRC violations in general because the via metal should still be covered by the metal above/below the via and avoid any min-step violations.

Fixed-name vias can cause problems in hierarchical flows as the fixed-names from different blocks may collide. Parameterized via-names can be freely changed by applications and avoid this problem. If the design has odd-mgrid width wires or pins, this option is normally a good choice.

If this attribute is selected as `auto`, it is `true` for OpenAccess design, `false` for non-OpenAccess design.

generate_special_via_use_track_offset

```
generate_special_via_use_track_offset {true | false}
```

Default: `false`

Data_type: bool, read/write

Uses track offset.

generate_special_via_use_trim_metal_enclosure

```
generate_special_via_use_trim_metal_enclosure {true | false}
```

Default: `false`

Data_type: bool, read/write

Forces the tool to use enclosures with the trimmed layer.

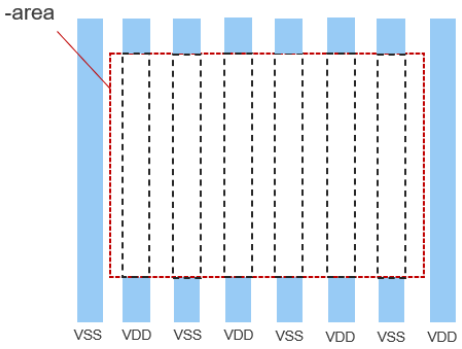
trim_pg

```
trim_pg
[-help]
[-area {x1, y1, x2, y2} | -candidate_file file_name]
-net {list_of_nets} {{-layer {layer_name1 [layer_name2]} -pattern {str1 [str2 … strN]} -type {stripe | via}} | -candidate_file
file_name}
[-exclude_local_grid {{lx1 ly1 ux1 uy1} {lx2 ly2 ux2 uy2}...} | -candidate_file file_name]
[-exclude {{lx1 ly1 ux1 uy1} {lx2 ly2 ux2 uy2}...} | -candidate_file file_name]
[-keep_dangling_via]
[-threshold threshold_value | -candidate_file file_name]
```

It is a semi-automated command that helps you trim the redundant PG stripes and vias using an existing power grid pattern based on IR drop analysis reports. This command eliminates the need to manually adjust the power structure to reduce the power grid pessimism for area/timing/power improvement in a design.

Parameters

`-help`	Prints a brief description that includes the type and default information for each `trim_pg` parameter. For a detailed description of the command and all its parameters, use the `man` command: `man trim_pg`
`-area {x1 y1 x2 y2}`	Specifies the trimming area. It defines the stripes and vias to be trimmed. The stripes overlapped with the box defined by `-area` are trimmed. *Default*: entire core area *Data_type*: rect, optional
`-candidate_file file_name`	Specifies the candidate filename. *Data_type*: string, optional
`-exclude {{lx1 ly1 ux1 uy1} {lx2 ly2 ux2 uy2}...}`	Excludes the box areas for the stripes and vias in which the part of stripes need not to be trimmed. This parameter is used along with the `-area` parameter to specify the exclusion areas. If a stripe partially overlaps with the exclusion area, the overlapping segment of the stripe is not trimmed. The exclusion box can partially overlap the `-area` box. *Data_type*: (rect)+, optional
`-exclude_local_grid {{lx1 ly1 ux1 uy1} {lx2 ly2 ux2 uy2}...}`	Excludes the areas in which the whole stripes are skipped for trimming. If a stripe is completely inside one of the `-exclude_local_grid box` (can touch the boundary), it is excluded from the trim candidates and is not counted when applying the trim pattern. It is useful when trimming is done on a large area but there are local stripes in this area, which could affect the global trim pattern. The exclusion box can partially overlap the `-area` box. *Data_type*: (rect)+, optional
`-keep_dangling_via`	Keeps the dangling vias while trimming stripes. By default, the dangling vias are removed. *Data_type*: bool, optional
`-layer {layer_name1 [layer_name2]}`	Specifies the metal layer names for stripe and via trimming. While trimming a stripe, specify `layer_name_1` on which the stripes are to be trimmed. While trimming via, specify `layer_name_2` between which the stack vias are to be trimmed. The first layer is used as the reference while applying the via trim pattern. *Data_type*: string, optional

`-net {list_of_nets}`	Specifies the list of power or ground nets to be trimmed. The following figure presents area trimmed by the `-net` parameter:  *Data_type*: string, required
`-pattern {str1 [str2 … strN]}`	Specifies the trimming pattern. While trimming a stripe, specify *str1*. The string is a sequence of “0” and “1” to indicate whether to trim or keep a stripe. While trimming a via, *str1* and multiple strings (if needed) define a multi-dimensional trim pattern. Each string is a sequence of “0” and “1” to indicate whether to trim or keep a stack via along the corresponding stripe on the reference layer. The reference layer is the first layer in the `-layer` parameter. *Data_type*: string, optional
`-threshold threshold_value`	Specifies the threshold value for stripe merging. The stripes with center-line spacing smaller than or equal to the threshold value are considered for trimming. *Default*: `0` *Data_type*: float, optional
`-type {stripe | via}`	Specifies the type of trimming either on the stripe of a layer or the stack via between two layers. *Data_type*: enum, optional

Examples

- The following command trims 50% of the `VDD` stripes on a `M6` metal layer inside a `{0 0 500 500}` box:
  ```
  trim_pg -net VDD -type stripe -layer M6 -area {0 0 500 500} -pattern 10
  ```
- The following command trims a via (stack) along with half of the `M9 VDD` stripes, and one-third of its stack vias between the `M9` and `M6` metal layers:
  ```
  trim_pg -net VDD -type via -layer {M9 M6} -area {0 0 500 500} -pattern {110 1}
  ```

Related Topics

Power Planning and Routing chapter in the *Innovus User Guide.*

update_power_vias

```
update_power_vias
[-help]
{-add_vias {0 | 1} | -update_vias {0 | 1} | -delete_vias {0 | 1}}
[-between_selected_wires {0 | 1} | -selected_blocks {0 | 1} | -selected_wires {0 | 1} | -area {{x1 y1 x2 y2} {x3 y3 x4 y4} ...}]
[-bottom_layer layername]
[-create_via_on_merged_target_on_layer layerName]
[-create_via_on_merged_target_within micron]
[-create_via_on_signal_pins {0 | 1}]
[-exclude_stack_vias {0 | 1}]
[-followpin_via_stapling {cutclass1 weight1 cutclass2 weight2}]
[-nets string]
[-orthogonal_only {0 | 1}]
[-selected_vias {0 | 1}]
[-split_long_via {threshold_value step_value offset_value height_value}]
[-split_vias {0 | 1}]
[-same_size_stack_vias {0 | 1}]
[-skip_via_on_wire_shape {[Blockring] [Stripe] [Followpin] [Corewire] [Blockwire] [Iowire] [Padring] [Ring] [Fillwire] [Noshape]}
[-skip_via_on_pin {[Pad] [Block] [Cover] [Standardcell] [ Physicalpin]}]
[-skip_via_on_wire_status {[routed] [fixed] [cover] [shield]}]
[-top_layer layername]
[-user_class subclass_string]
[{-via_scale_height integer -via_scale_width integer
    | -via_rows integer -via_columns integer
    | -via_height real_number -via_width real_number}]
[-via_using_exact_crossover_size {0 | 1}
```

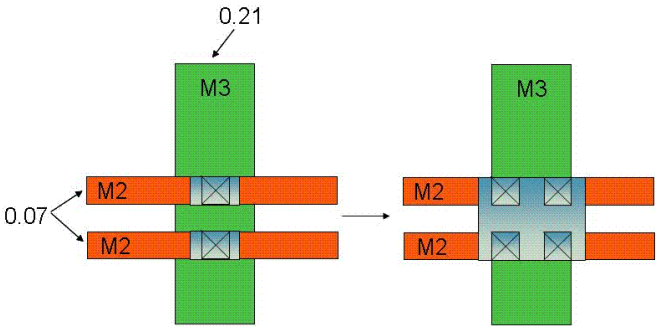
Adds power vias to the design or performs one of the following actions to existing power vias.

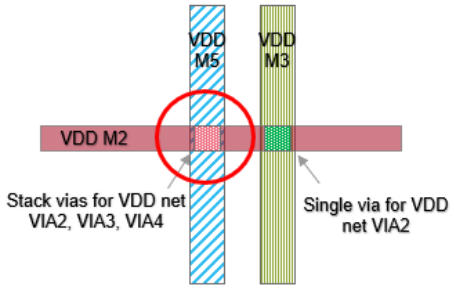
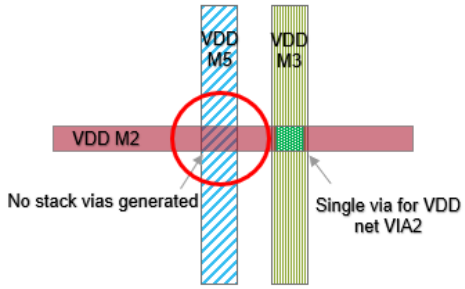
- Modifies
- Deletes
- Fixes vias that violate the LEF `MINIMUMCUT` rule

By default, the action you specify applies to all vias in the design.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `update_power_vias` parameter. For a detailed description of the command and all its parameters, use the `man` command: `man update_power_vias`
`-add_vias {0 | 1}`	Adds power vias in your design. *Default:* `0` *Data_type*: bool, optional
`-area {{x1 y1 x2 y2} {x3 y3 x4 y4} ...}`	Adds power vias on power wires/pins overlapped with the specified area regions enclosed by the specified coordinates; modifies or deletes power vias within the specified area regions. *Note:* If you need to add only the power vias within the specified area, run `set_db -area_only 1` before `update_power_vias`. *Data_type*: string, optional

`-between_selected_wires {0 | 1}`	Adds, modifies, or deletes power vias between selected wires or pins in your design. You must select these interactively in the design display area. *Default:* `0` *Data_type*: bool, optional
`-bottom_layer layername`	Specifies the lowest layer to which vias can connect. *Default:* The lowest metal layer in the design. *Data_type*: string, optional
`-create_via_on_merged_target_within` *micron*	
	Helps `viaGen` merge the targets within the halo distance on the specified layer. *Data_type*: float, optional
`-create_via_on_merged_target_on_layer` *layerName*	
	Helps `viaGen` add a virtual target internally, ignore the original targets, and drop vias to the virtual target instead.  *Data_type*: string, optional
`-create_via_on_signal_pins {0 | 1}`	
	Allows generation of vias from special wire to signal pins when set to `1`. `update_power_vias` drop vias between special route and PG/signal pin, including the flipchip route. *Default*: `0` **Note:** This parameter can be used with the `-area` parameter to specify the area for adding vias. It mainly occurs at the Tie high/low cell pins. *Data_type*: bool, optional
`-delete_vias {0 | 1}`	Deletes power vias from your design. *Default:* `0` *Data_type*: bool, optional

`-exclude_stack_vias {0 | 1}`	Avoids generation of stack vias between two non-adjacent layers to handle the intersection cases of cross multiple layers. See the examples below to understand the behavior of this parameter with different values: • `update_power_vias -exclude_stack_vias 0 -add_vias 1 …`  • `update_power_vias -exclude_stack_vias 1 -add_vias 1 …`  *Default:* `0` *Data_type*: bool, optional
`-followpin_via_stapling {cutclass1 weight1 cutclass2 weight2}`	Defines two cut classes to be tried in priority in the followpin via stapling to maximize the effective number of cuts. The weight value is for via insertion ratio calculation. *Data_type*: string, optional
`-update_vias`	Modifies vias in the design. *Data_type*: bool, optional
`-nets string`	Specifies the nets to drop the vias. *Data_type*: string, optional
`-orthogonal_only {0 | 1}`	Creates vias *only* for orthogonal wires and pin intersections. *Default*: If you do not specify this parameter or if you specify it with a value of `1`, the software creates vias for orthogonal wires and pin intersections. *Data_type*: bool, optional
`-same_size_stack_vias {0 | 1}`	If set to `1`, the software always generates same-sized stacked vias. If set to `0`, the software trims the vias as needed to meet the DRC rules. *Default:* `0` *Data_type*: bool, optional
`-selected_blocks {0 | 1}`	Specifies if vias in selected blocks only should be looked up. *Data_type*: bool, optional

`-selected_vias {0 | 1}`	Modifies/deletes the selected vias. *Data_type*: bool, optional
`-selected_wires {0 | 1}`	Specifies to drop vias between selected wires and all same net objects across the selected wires. *Data_type*: bool, optional
`-skip_via_on_pin {[Pad] [Block] [Cover] [Standardcell] [ Physicalpin]}`	
	Prevents vias from being generated on the specified types of pins. If you specify -skip_via_on_pin {} , the power planning software makes connections to all five categories of pins. *Default:* If you do not specify this parameter, vias are generated for block pins, pad pins, and cover macro pins, but are not generated for standard cell pins. *Data_type*: (enum)+, optional
`-skip_via_on_wire_shape {[Blockring] [Stripe] [Followpin] [Corewire] [Blockwire] [Iowire] [Padring] [Ring] [Fillwire] [Noshape]}`	
	Prevents vias from being generated for the specified wire shapes. *Default:* If you do not specify this parameter, vias can be generated for all wire shapes. *Data_type*: (enum)+, optional
`-skip_via_on_wire_status {[routed] [fixed] [cover] [shield]}`	
	Prevents vias between wires with routed, cover, fixed, shield status and other power wires. *Default:* If you do not specify this parameter, vias can be generated for any wire status. *Data_type*: (enum)+, optional
`-split_long_via {`*`threshold_value step_value offset_value height_value`*`}`	
	Splits vias longer than the specified length (`threshold_value`, in micrometers) into smaller vias with specified center-to-center spacing (`step_value`, in micrometers), left/bottom end offset (`offset_value`, in micrometers) and vertical/horizontal height (`height_value`, in micrometers) values. This is independent of the option `-split_via`. *Default*: If you do not specify this parameter, the software does not split vias. **Note:** If the `offset_value` is given as negative, the vias will be placed symmetrically on both ends. The default value of `offset_value` is `-1`. *Data_type*: string, optional
`-split_vias {0 | 1}`	Specifies whether two or more partial vias can be created at the crossover between a stripe and any target (pin or wire) if an obstruction (macro obstruction; wire or pin on a different net) prevents the creation of a full-size via. This can be useful, for example, if several metal blockages partially obstruct the area where a full-size via would normally be placed. This parameter permits the creation of partial vias that would not violate the process rules, in the open areas. A value of 1 allows multiple partial vias to be created. A value of 0 prevents multiple partial vias from being created. *Default:* `0` *Data_type*: bool, optional
`-top_layer` *`layername`*	Specifies the highest layer to which vias can connect. *Default:* The highest metal layer in the design. *Data_type*: string, optional
`-user_class` *`subclass_string`*	Indicates the user defined attribute subclass for wires to route. *Data_type*: string, optional

`-via_height` *real_number* `-via_width` *real_number*	Specify the absolute height and width of the added or modified power vias. You can use these parameters only if you choose the `-add_vias` or `-update_vias` parameter. Default: 0.0 *Data_type*: float, optional
`-via_rows` *integer* `-via_columns` *integer*	Specifies the number of cuts to make in the power vias. You can use these parameters only if you choose the `-add_vias` or `-update_vias` parameter. Default: 0
`-via_scale_height` *integer* `-via_scale_width` *integer*	Specify the percentage to scale the power via dimensions. You can use these parameters only if you choose the `-update_vias` or `-add_vias` parameter. *Data_type*: float, optional
`-via_using_exact_crossover_size {0| 1}`	Specifies whether to generate partial vias of the exact crossover size. A value of 1 allows partial vias to be generated. A value of 0 prevents partial vias, and the software always generates full-size vias. **Note**: This option does not apply when both of the objects at the crossover are rings or stripes, or at the crossover of a ring and stripe. In these cases, the software always generates a full-size via array. *Default:* `1` *Data_type*: bool, optional

Example

The following command creates a power via between selected wires with a cut pattern of 3 by 3:

```
update_power_vias -add_vias 1 -between_selected_wires 1 -via_rows 3 -via_columns 3
```

Related Topics

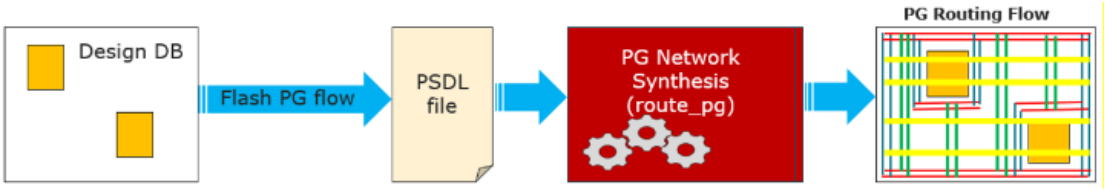
- `set_preference`
- select_routes
- `add_split_power_vias`
- `check_power_vias`
- Power Planning and Routing chapter in the *Innovus User Guide.*

route_pg

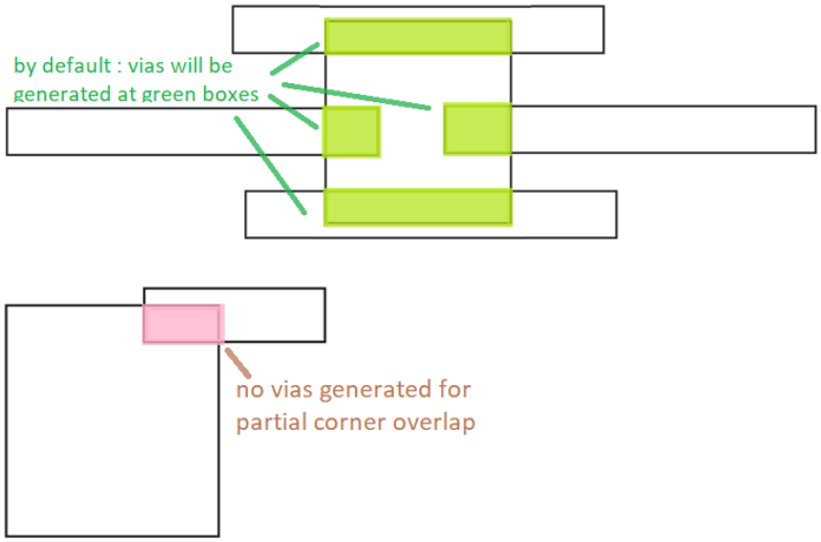
```
  route_pg
[-help]
[-check_psdl_only]
[-delete_floating]
[-fast]
[-psdl_file file_name]
[-skip_block_pin]
[-skip_partial_overlap_via]
[-skip_via]
```

Uses the Innovus database, builds the region with the help of PSDL, where the PG structure is to be created, and parses it. This command also uses all the syntax in PSDL to build the PG structure.

The following figure presents the process of developing a PG routing flow using the `route_pg` command:



Parameters

`-help`	Outputs a brief description that includes the type and default information for each `route_pg` parameter. For a detailed description of the command and all its parameters, use the `man` command: `man route_pg`
`-check_psdl_only`	Performs sanity check only on the PSDL input file. *Data_type*: bool, optional
`-delete_floating`	Removes floating wires. *Data_type*: bool, optional
`-fast`	Generates PG routing flow in the Fast PG mode. *Data_type*: bool, optional
`-psdl_file` *file_name*	Specifies the PSDL input file. *Data_type*: string, optional
`-skip_block_pin`	Skips the formation of connection to the block pins. *Data_type*: bool, optional
`-skip_partial_overlap_via`	Skips building of via on partial overlap intersection. See the example below:  *Data_type*: bool, optional

`-skip_via`	Skips building via object. *Data_type*: bool, optional

Examples

- The following command checks the `-psdl_file` syntax only (without running the `route_pg` command):
  `route_pg -fast -psdl_file filename -check_psdl_only`
- The following command removes a floating wire:
  `route_pg -fast -psdl_file psdl_file -delete_floating`
- The following command does not generate via while running the `route_pg` command:
  `route_pg -fast -psdl_file psdl_file -skip_via`
- The following command skips partial generated vias:
  `route_pg -fast -psdl_file psdl_file -skip_partial_overlap_via`

Related Information

- The PG Network Synthesis chapter of Innovus User Guide.

29

Power Route Commands and Attributes

- fix_via
- report_power_route
- route_special
- route_special Category Attributes

fix_via

```
fix_via
[-help]
{-short | -min_cut | -min_step}
[-area]
```

Fixes the following types of violations in special net vias: short, mincut, and minstep. If the via cannot be replaced, the violation marker is not removed.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `fix_via` parameter. For a detailed description of the command and all its parameters, use the `man` command: `man fix_via`
`-area {x1 y1 x2 y2}`	
	Specifies the rectangular area. Thus, the command applies to the violations within the area only. The area option should be compatible with all existing options and supports all violations that the command can support at the moment. Specifying the area improves the run-time for large design with significant number of violations. `x1`, `y1` are the coordinates for lower left corner of the area. `x2`, `y2` are the coordinates for the upper right corner of the area. By default, this option is off and with no default value for any of its argument. *Data_type*: rect, optional
`-short`	Fixes vias which have a short violation. *Data_type*: bool, required
`-min_cut`	Replaces power vias flagged by a violation marker due to a violation of the LEF `MINIMUMCUT` rule. The via is replaced with a via that contains the minimum number of cuts based on the LEF rule, and the violation marker is removed. *Data_type*: bool, required
`-min_step`	Replaces power vias flagged by a violation marker due to a violation of the LEF `MINIMUMSTEP` rule. The via is replaced with a via that contains the minimum number of steps based on the LEF rule, and the violation marker is removed. *Data_type*: bool, required

Related Topics

Power Planning and Routing chapter in the *Innovus User Guide.*

report_power_route

```
report_power_route
[-help]
[-followpin_cut_ratio {cutClass1 weight1 cutClass2 weight2}]
[-net string]
```

Reports the cut ratio when two types of cuts are inserted between two parallel followpin rails with `update_power_vias -followpin_via_stapling` to maximize the effective number of cuts. The ratio is calculated by dividing the sum of weighted cut count of each cut class by theoretical cut count, if the smallest via is inserted along with the power rail with minimum spacing.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `report_power_route` parameter. For a detailed description of the command and all its parameters, use the man command: `man report_power_route`
`-followpin_cut_ratio {cutClass1 weight1 cutClass2 weight2}`	
	Specifies the names and weights of the two types of the cut classes. *Data_type*: string, optional
`-net string`	Specifies the followpin net names to be reported. If this parameter is not specified, the report is calculated for all the followpin nets. *Data_type*: string, optional

Example

The following command specifies the weights for the `VDOUBLE` and `VSINGLE` cut classes and calculates the reports for the `VSS` followpin net:

```
report_power_route -followpin_cut_ratio {VDOUBLE 1.8 VSINGLE 1} -net VSS
```

Related Topics

Power Planning and Routing chapter in the *Innovus User Guide.*

route_special

```
  route_special
[-help]
[-allow_jogging {0 | 1}]
[-allow_layer_change {0 | 1}]
[-area {x1 y1 x2 y2}]
[-inst {names_of_blocks}]
[-block_pin {use_lef | all | on_boundary | left_boundary | right_boundary |
top_boundary | bottom_boundary | abut_pins}]
[-block_pin_layer_range {minLayerName maxLayerName}]
[-block_pin_width_range {min max}]
[-block_pin_target {nearest_target | boundary_with_pin | farthest_pad_ring} |
{[block_ring] [pad_ring] [ring] [stripe] [ring_pin] [block_pin]}]
[-connect {[block_pin] [core_pin] [pad_pin] [pad_ring] [floating_stripe]
[secondary_power_pin]}]
[-connect_aligned_block_and_pad_pin {block_pin_as_target | pad_pin_as_target}]
[-connect_in_area]
[-core_pin_check_stdcell_geometry]
[-core_pin_layer layerNumList]
[-core_pin_target {{[block_ring] [ring] [stripe] [pad_ring] [ring_pin] [block_pin]}|{
first_after_row_end | boundary_with_pin | farthest_pad_ring | none }}]
[-core_pin_width real]
[-crossover_via_layer_range {bot top}]
[-delete_existing_routes]
[-detailed_log]
[-floating_stripe_target {[block_ring] [ring] [stripe] [pad_ring] [ring_pin]
[block_pin] [followpin]}]
[-layer_change_range {topLayerName bottomLayerName}]
[-nets {names}]
[-all_overlapped_block_pins]
[-pad_pin_layer_range {minLayerName maxLayerName}]
[-pad_pin_width real]
[-pad_pin_port_connect {[one_port | all_port ][ one_geom |all_geom] | prefer_layer}
[-pad_ring_width real]
[-pad_ring_layer {layerNumList}]
[-power_domains [powerDomainName1 powerDomainName2 …]]
[-secondary_pin_net {list_of_nets}]
[-secondary_pin_rail_vertical_stripe_grid {layerName width pitch}]
[-stripe_layer_range {minLayerName maxLayerName}]
[-target_obj_list_file filename]
[-target_via_layer_range {bot top}]
```

```
[-user_class subclass_string]
[-pad_cell_skip_routing_on_pad_side {top bottom left right}]
[-pad_pin_target {nearest_target | {[block_ring] [ring] [stripe] [ring_pin] [block_pin]
[followpin]}}]
[-secondary_pin_rail_layer]
```

Routes power structures. Use this command after creating power rings and power stripes. Depending on the parameters you use, it can do any or all of the following:

- Create pad rings
- Connect pins on specified nets on the blocks
- Connect pads to nearby rings or stripes
- Extend unconnected stripes to make a connection
- Route standard cell pins

Note: To undo the `route_special` command, use the `defOut` command to save the design database before you issue the `route_special` command. Then you can use the `read_def` command to restore the design to its state before running `route_special`.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `route_special` parameter. For a detailed description of the command and all its parameters, use the `man` command: `man route_special`
`-allow_layer_change {0 | 1}`	Allows connections to targets on different layers. If the metal1 preferred routing direction is vertical and standard cells have their VDD and VSS pins on layer metal1, route_special will change routing layers to continue extending the followpin connections to core rings by this option. *Data_type*: bool, optional

`-allow_jogging {0 | 1)`	Specifies that jogs are allowed during routing to avoid DRC violations. **Note:** You can use `-allow_jogging 1` and `-connect pad_ring` together to enable jog connection in pad ring connection. *Data_type*: bool, optional
`-area {`*`x1 y1 x2 y2`*`}`	Specifies the coordinates for the area in which power is to be routed. *Default:* If you do not specify this parameter, the entire design is routed. If you specify this parameter, existing routes are always preserved, even if you also specify the `-delete_existing_routes` parameter. *Data_type*: rect, optional
`-inst {`*`names_of_blocks`*`}`	Specifies the blocks to connect. **Note:** This parameter replaces the `-blocks` parameter. *Data_type*: string, optional
`-block_pin {use_lef | all | on_boundary | left_boundary | right_boundary | top_boundary | bottom_boundary | abut_pins}`	

	Specifies which block pins to connect. Use one of the following parameter values: • `abut_pins`: Pins are abutted. • `all`: All pins on all ports are connected, regardless of how pins are specified in the LEF file. • `bottom_boundary`: Only pins on the bottom boundary are connected. • `on_boundary`: Connects *only* pins that abut block boundaries. • `top_boundary`: Only pins on the top boundary are connected. • `use_lef`: Pins are connected exactly as specified in the LEF file. For example, if the LEF file has multiple ports with one geometry connecting to each port, then one pin per port is connected. *Data_type*: (enum)+, optional
`-block_pin_layer_range {`*`minLayerName maxLayerName`*`}`	Specifies the layers on which the tool can make connections, between the bottom-most and top-most layers. A value of `1` indicates the lowest metal later, `2` indicates the next layer up, and so forth. • `maxLayerName`: Specifies the top-most block pin layer to connect to. • `minLayerName`: Specifies the bottom-most block pin layer to connect to. *Data_type*: string, optional
`-block_pin_target {nearest_target | boundary_with_pin | farthest_pad_ring} | {[block_ring] [pad_ring] [ring] [stripe] [ring_pin] [block_pin]}}`	

	Extends the block pin to targets of specified type. Multiple target types are allowed. The block pin will be open if no specified target is found: • `nearest_target`: Extends the block pin to the nearest legal target. • `boundary_with_pin`: Extends the block pin to the design boundary and creates a new power pin along the design boundary. Any overlaps with existing I/O pins at the design boundary are flagged as violations after the extension. • `fathestpad_ring` : Extends the block pin to the last pad ring that already exists on top of the I/O pad cells. • `block_ring pad_ring ring stripe ring_pin block_pin` **Note:** • Specify either {`nearest_target`}, `{boundary_with_pin}`, `{farthest_pad_ring}`, or a value from the following for which multiple keywords are allowed: {`block_ring pad_ring ring stripe ring_pin block_pin`} • Shielding wire is not regarded as the target of the block pin. *Data_type*: (enum)+, optional

`-block_pin_width_range {min max}`	Specifies the allowable pin widths, between the maximum and minimum widths given. Specifies the maximum block pin width to connect to. *Default:* If you do not specify this parameter, the software uses a value of `100`. Specifies the minimum block pin width to connect to. *Default*: `0.0` *Data_type*: string, optional
`-connect {[block_pin] [core_pin] [pad_pin] [pad_ring] [floating_stripe] [secondary_power_pin] }`	
	Connects the specified objects to rings and stripes. If `-connect` is not provided, route_special connects `pad_pin`, `pad_ring`, `core_pin`, `block_pin` and `floating_stripe` by default. If `-connect {}` is provided, route_special does nothing and exits. To connect `secondary_power_pin`, you must explicitly use the combination of `-connect {secondary_power_pin}` and `-secondary_pin_net {netname}`. **Note:** You can use `-allow_jogging 1` and `-connect pad_ring` together to enable jog connection in pad ring connection. *Data_type*: (enum)+, optional
`-connect_aligned_block_and_pad_pin {block_pin_as_target | pad_pin_as_target}`	

	Specifies whether block or pad pins can be connection targets for aligned pad or block pins. • `block_pin_as_target`: Specifies that a block pin can be considered a connection target for a pad pin if the block pin is aligned with the pad pin. • `pad_pin_as_target`: Specifies that a pad pin can be considered a connection target for a block pin if the pad pin is aligned with the block pin. *Data_type*: enum, optional
`-connect_in_area`	Specifies that connections from all sources within the specified area can connect only to targets that are also inside that area. **Note:** This parameter is only available if you also specify the `-area` parameter. *Default:* If you do not specify this option the software makes connections from all sources within the specified area to targets both inside and outside the specified area. *Data_type*: bool, optional
`-core_pin_check_stdcell_geometry`	Checks the standard cell geometries in the design for any spacing violations between the power rail to power stripe vias and cell blockages to repair any DRC violations. When detected, Innovus repairs these violations by trimming the via arrays as needed. Use this parameter after routing the power structures. *Default*: off *Data_type*: bool, optional

`-core_pin_layer {`*`layerNumList`*`}`	Connects power pins only on the specified layers. A value of `1` indicates the lowest metal later, `2` indicates the next layer up, and so forth. If you specify more than one layer, you must enclose the list in curly braces. *Default:* If you do not specify this parameter, only pins on metal1 or metal2 will be connected. **Note:** Recommend to use LEF layer name. *Data_type*: string, optional
`-core_pin_target {{[block_ring] [ring] [stripe] [pad_ring] [ring_pin] [block_pin]}|{ first_after_row_end | boundary_with_pin | farthest_pad_ring | none }}`	
	Specifies the extension target for followpins. • `first_after_row_end`: Extends the standard cell pin to the first ring or stripe outside of the row. • `boundary_with_pin`: Extends the standard cell pin to the design boundary and creates a new power pin along the design boundary. Any overlaps with existing I/O pins at the design boundary are flagged as violations after the extension. • `farthest_pad_ring`: Extends the standard cell pin to the last pad ring that already exists on top of the I/O pad cells. • `none`: Unconnected standard cell pins are not extended, and connections may be left open. • `block_ring ring stripe pad_ring ring_pin block_pin` **Note:** Followpin wire can be targets for stripe and secondary pin connection, but not for followpin connection. *Default:* `first_after_row_end`. *Data_type*: (enum)+, optional

`-core_pin_width` *real*	If `route_special -core_pin_width` is run prior to placement, then the standard cell rails are created with the specified width. If it is run after the placement, *route_special* will only route pins that have the specified width. *Data_type*: float, optional
`-crossover_via_layer_range {`*bot top*`}`	Specifies the highest and lowest layer that can be used for via stacking at the crossover point between power structures. **Note:** This parameter does not apply to T-pattern connections. *Default:* If you do not specify this parameter, all layers are used. *Data_type*: string, optional
`-delete_existing_routes`	Specifies that the software remove existing connections when you use the `route_special` command multiple times. *Default:* The software maintains existing connections each time you issue the `route_special` command. **Note:** If you specify the `-area` parameter, existing routes are always preserved, even if you also specify the `-delete_existing_routes` parameter. *Data_type*: bool, optional
`-detailed_log`	Prints the detailed log file. *Data_type*: bool, optional
`-floating_stripe_target {[block_ring] [ring] [stripe] [pad_ring] [ring_pin] [block_pin] [followpin]}`	
	Specifies the target shape for floating stripes. *Default:* `{block_ring ring stripe pad_ring ring_pin block_pin followpin}` *Data_type*: (enum)+, optional

`-layer_change_range {`*topLayerName bottomLayerName*`}`	Allows routing between the specified bottom-most and top-most layer, inclusive. *Default*: If you do not specify this parameter, the tool can use any layer. *Data_type*: string, optional
`-nets {`*names*`}`	Specifies the names of the nets to connect. If you connect to more than one net, you must enclose the net names in curly braces and separate the names with a space. This parameter supports wildcard characters (like * and ?), when the `route_special_signal_pin_as_pg` attribute is set to `true`. *Default:* If you do not specify this parameter, the software connects all power and ground nets in the design. *Data_type*: string, optional
`-all_overlapped_block_pins`	Specifies that all pins (including those that overlap) should be routed. **Note:** This parameter is effective *only* if `route_special -block_pin all` is specified. *Default*: If you omit this parameter, if multiple pins overlap either fully or partially, only one will be routed. *Data_type*: bool, optional

`-pad_cell_skip_routing_on_pad_side {top bottom left right}`	Specifies the pad pins of the pads should not be connected. • `bottom`: Specifies the pad pins on the bottom side of the pads should not be connected.. • `left`: Specifies the pad pins on the left side of the pads should not be connected. • `right`: Specifies that power pads along the right side of the chip should not be connected. • `top`: Specifies that power pads along the top side of the chip should not be connected. *Data_type*: (enum)+, optional
`-pad_pin_layer_range {minLayerName maxLayerName}`	Specifies that routing is allowed between the bottom-most and top-most pad pin layers, inclusive. • `maxLayerName`: Specifies the top-most pad pin layer allowed for connection. • `minLayerName`: Specifies the bottom-most pad pin layer allowed for connection. *Data_type*: string, optional
`-pad_pin_port_connect {[one_port | all_port ][ one_geom |all_geom] | prefer_layer}`	

	Specifies connections to pad pin ports. • `one_port`: Routes to all geometries of the same port for each pad pin. • `all_port`: Routes to all ports. • `one_geom`: Routes to one geometry per port only. • `all_geom`: Routes to only one port of a pad pin if multiple ports are defined in the LEF file. • `prefer_layer`: Routes to all geometries on the same port, but in the preferred routing direction only. *Data_type*: (enum)+, optional
`-pad_pin_target {nearest_target | {[block_ring] [ring] [stripe] [ring_pin] [block_pin] [followpin]}}`	
	Specifies the target type.Extends the pad pin to targets of specified type. Multiple target types are allowed. The pad pin will be open if no specified target found. • `nearest_target`: Extends the pad pin to the nearest legal target. • `block_ring ring stripe ring_pin block_pin`: *Default*: `nearest_target` *Data_type*: (enum)+, optional
`-pad_pin_width` *real*	Routes only pad pins that have the specified width. If no width is specified, the software automatically calculates the width of the pad pins to use. It is usually not necessary to specify a pin width. *Data_type*: float, optional

`-pad_ring_layer` *layerNumber*	Specifies which routing layer to use. A value of `1` indicates the lowest metal later, `2` indicates the next layer up, and so forth. The default is all layers. **Note:** Recommend to use LEF layer name. *Data_type*: string, optional
`-pad_ring_width` *real*	Specifies the width of pins to connect with. By default, the software automatically calculates this width. *Data_type*: float, optional
`-power_domains [`*powerDomainName1 powerDomainName2 …*`]`	
	Restricts `route_special` activity to the specified power domains. *Default*: All power domains **Note:** The `-delete_existing_routes` parameter will ignore the `-power_domains` parameter while removing existing connections. To remove special nets in a power domain, run `editDelete` manually before `route_special`. *Data_type*: string, optional
`-secondary_pin_net {`*list_of_nets*`}`	Connects voltage level shifter pins on the specified nets to the closest segment of the ring around the power domain. This option is available only if a power domain is defined. *Data_type*: string, optional
`-secondary_pin_rail_layer` *layerName*	Specifies connections to secondary pin rail layers. *Data_type*: string, optional
`-secondary_pin_rail_vertical_stripe_grid {`*layerName width pitch*`}`	

	Specifies connections to secondary power pins. • `topLayerNum`: Specifies the top-most stripe layer on which the tool can route secondary power pins. • `width`: Specifies the width (in user units) of stripes that connect to secondary power pins. • `pitch`: Specifies the pitch (in user units) of stripes that connect to secondary power pins. *Data_type*: string, optional
`-stripe_layer_range {`*minLayerName maxLayerName*`}`	
	Specifies the floating stripe layer range allow to be connected by `route_special`. You need to specify the following values with this parameter: • `maxLayerName`: Specifies the top-most floating stripe layer allowed to be connected by route_special. • `minLayerName`: Specifies the bottom-most floating stripe layer allowed to be connected by route_special. *Data_type*: string, optional
`-target_obj_list_file` *filename*	Specifies a `.obj` file that contains a list of objects that the route_special software is to either consider as a target or remove from the list of available targets. *Data_type*: string, optional
`-target_via_layer_range {`*bot top*`}`	Specifies the highest and lowest layer that can be used for via stacking at a target. *Default:* If you do not specify this parameter, all layers are used. *Data_type*: string, optional

`-user_class` *`subclass_string`*	Displays the user defined attribute attached to followpin, blockwire, core wires, iowires and pad_ring. *Data_type*: string, optional

Related Topics

Power Planning and Routing chapter in the *Innovus User Guide.*

route_special Category Attributes

The root attributes in the `route_special` category are:

- route_special_allow_non_preferred_direction_route
- route_special_avoid_over_core_row_layer
- route_special_block_pin_connect_ring_pin_corners
- route_special_block_pin_route_with_pin_width
- route_special_connect_broken_core_pin
- route_special_core_pin_ignore_obs
- route_special_core_pin_length
- route_special_core_pin_length_as_inst
- route_special_core_pin_max_via_scale
- route_special_core_pin_merge_limit
- route_special_core_pin_refer_to_follow_pin
- route_special_core_pin_reference_macro
- route_special_core_pin_site_rail_width
- route_special_core_pin_snap_to
- route_special_core_pin_stop_route
- route_special_endcap_as_core
- route_special_extend_nearest_target
- route_special_jog_threshold_ratio
- route_special_layer_preferred_direction_cost
- route_special_layer_non_preferred_direction_cost
- route_special_pad_pin_min_via_size
- route_special_pad_pin_split
- route_special_pad_ring_use_lef
- route_special_pg_pin_as_signal

- route_special_secondary_pin_max_gap
- route_special_secondary_pin_rail_width
- route_special_signal_pin_as_pg
- route_special_split_long_via
- route_special_target_number
- route_special_target_search_distance
- route_special_time_limit
- route_special_via_connect_to_shape
- route_special_via_through_to_closest_ring
- route_special_welltap_as_endcap

To set the value of any root attribute, use the `set_db` command:

```
set_db attribute_name value
```

To get the current value of any root attribute, use the `get_db` command:

```
get_db attribute_name
```

To get the current value of all attributes in the `route_special` category, use the following command:

```
get_db -category route_special
```

route_special_allow_non_preferred_direction_route

```
route_special_allow_non_preferred_direction_route {true | false}
```

Default: `false`

Data_type: bool, read/write

Allows wrong way routing.

Note: If the metal1 preferred routing direction is vertical and standard cells have their VDD and VSS pins on layer metal1, sroute will route metal1 wires on horizontal direction to continue extending the followpin connections to core rings by this option.

route_special_avoid_over_core_row_layer

```
route_special_avoid_over_core_row_layer layer_name
```

Default: `max layer of standard cell pin`

Data_type: string, read/write

Specifies the number of layers to avoid routing of block pin and pad pin over the core row area.

route_special_block_pin_connect_ring_pin_corners

`route_special_block_pin_connect_ring_pin_corners {true | false}`

Default: `false`

Data_type: bool, read/write

Specifies that the block wire used for a connection is to have the same width as the pin from which it connects.

route_special_block_pin_route_with_pin_width

`route_special_block_pin_route_with_pin_width {true | false}`

Default: `false`

Data_type: bool, read/write

Specifies that the block wire used for a connection is to have the same width as the pin from which it connects.

route_special_connect_broken_core_pin

`route_special_connect_broken_core_pin {true | false}`

Default: `false`

Data_type: bool, read/write

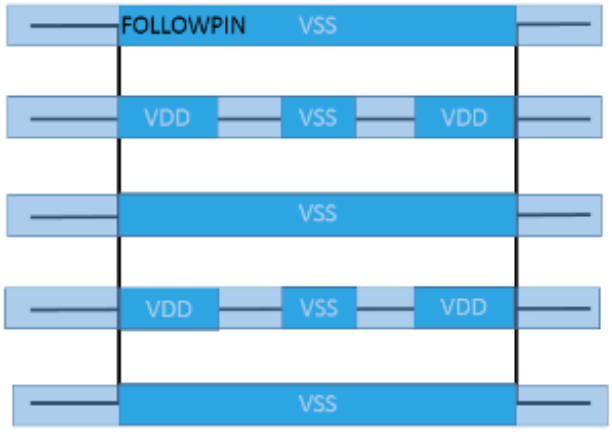
When set to true, `route_special` checks the signal pin, second power pin, and obs, and then breaks the followpin to avoid violations. When set to false, `route_special` does not check internal geometries and does not break the followpin.

Note that for ABUTMENT standardcell pins, followpins will go through the pin; for FEEDTHRU standard cell pins, followpins will break at the at the pin edge on the standard cell boundary. Refer to LEF/DEF Language Reference for more details.

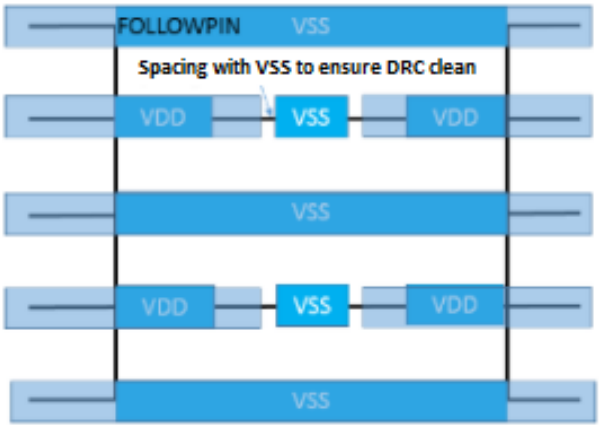
See the examples below for understanding the usage of this argument:

For ABUTMENT Standard Cell Pins:

```
set_db route_special_connect_broken_core_pin  false
```

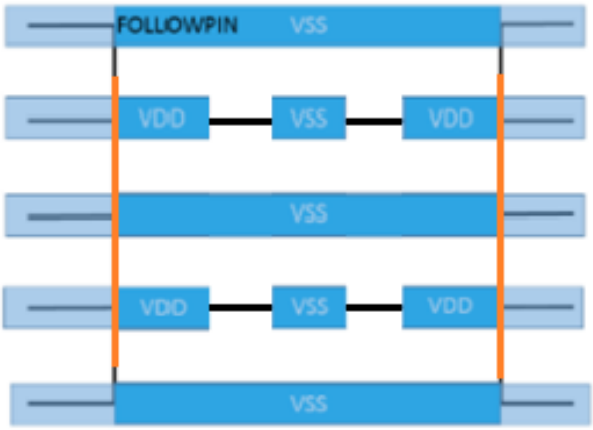



```
set_db route_special_connect_broken_core_pin true
```

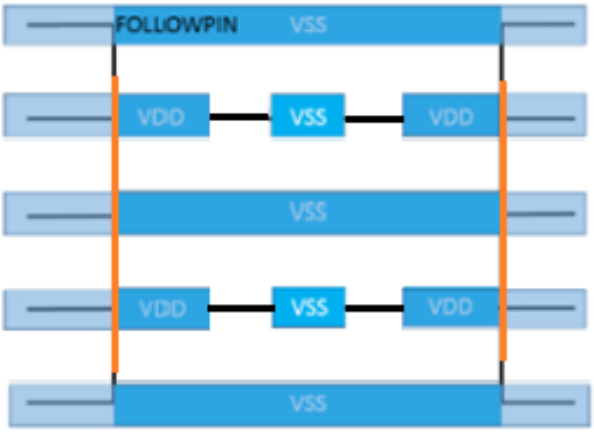


For FEEDTHRU Standard Cell Pins:

```
set_db route_special_connect_broken_core_pin false
```



```
set_db route_special_connect_broken_core_pin true
```



route_special_core_pin_ignore_obs

```
route_special_core_pin_ignore_obs {none | placement_blockage | overlap_obs |
block_halo}
```

Default: `none`

Data_type: enum* {none placement_blockage overlap_obs block_halo}, read/write

Ignores specified blockage when connecting the follow pin.

route_special_core_pin_length

```
route_special_core_pin_length value
```

Default: `0`

Data_type: double, read/write

Connects core pins of specified length. The option is of type *float*.

route_special_core_pin_length_as_inst

```
route_special_core_pin_length_as_inst {true | false}
```

Default: `false`

Data_type: bool, read/write

Connects core pins whose length equals or is greater than the length of instance.

route_special_core_pin_max_via_scale

```
route_special_core_pin_max_via_scale widthPct heightPct
```

Default: If you do not specify this parameter, the software uses a value of 100, which specifies a full-size via.

Data_type: string, read/write

Controls the length and width of vias at the crossover of core pin rails and stripes. Specify this value as percentages of the width of the via at the top of the stack. Smaller vias leave room for more routing tracks on layers between the vias.

route_special_core_pin_merge_limit

```
route_special_core_pin_merge_limit value
```

Default: `0`

Data_type: double, read/write

Joins follow pin rail gaps smaller than this variable, bigger gaps will remain open. The option is of type *float*.

route_special_core_pin_refer_to_follow_pin

```
route_special_core_pin_refer_to_follow_pin {true | false}
```

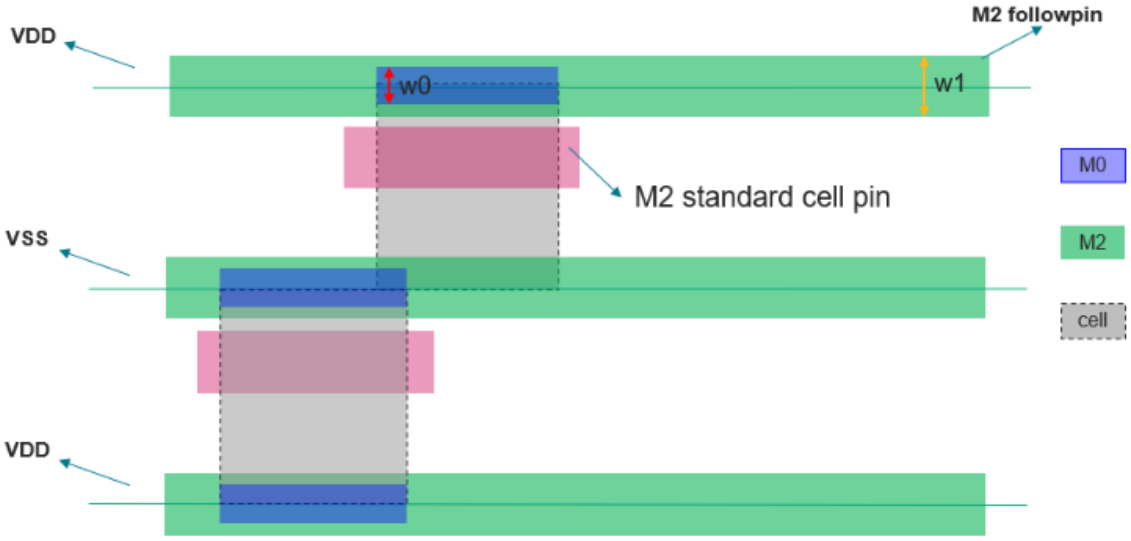
Default: `false`

Data_type: bool, read/write

If set to true, `route_special` generates followpins based on the standard followpins, whether or not there are standard cell pins on the defined metal layer. Standard followpins refer to the followpins on the first routing layer. This parameter can be used with any set of layers with same direction. Following is a usage example of this parameter:

```
set_db -route_special_core_pin_refer_to_follow_pin 1 …
route_special -core_pin_layer M2 -core_pin_width w1 … #w0 is the standard followpin's
width
```

Following is the output of the above command:



route_special_core_pin_reference_macro

`route_special_core_pin_reference_macro {macro1 macro2 ...}`

Default: `""`

Data_type: string, read/write

Uses specified macros as references for followpin creation. If you specify multiple macros for a row height, the software picks the first macro as reference.

route_special_core_pin_site_rail_width

`route_special_core_pin_site_rail_width {{site1 layer11 Width11 layer12 Width12 …} {site2 layer21 Width21 layer22 Width22 …} …}`

Default: `""`

Data_type: string, read/write

Defines the followpin layer width for the specified site rail for followpin generation. The followpin width defined by this attribute has higher priority than the width of the same layer defined in the macro LEF, but lower than that defined by the `route_special -core_pin_width` parameter.

Following are the values used with this attribute:

- `site`: Specifies the site name defined in the LEF file.
- `layer`: Specifies the name of the layer defined in the LEF file.

- `width`: Specifies the followpin width on the top and bottom sides of the defined site.

The following example of this attribute defines followpin layer width for generating followpins:

```
set_db route_special_core_pin_site_rail_width  {{site1 layer1 w1} {site2 layer1 w2}}
```

Following is the output of the above command:

route_special_core_pin_snap_to

```
route_special_core_pin_snap_to {M1_pin | opt_routing_track | grid | half_grid}
```

Default: `M1_pin`

Data_type: enum* {m1_pin opt_routing_track grid half_grid}, read/write

Snaps metal2 followpins in the location to save routing tracks in a preventive manner. When set to:

- `M1_pin`: Generated metal2 followpins are snapped to metal1 stdcell pin center.
- `opt_routing_track`: Generated metal2 followpins are snapped to track or half track to save routing tracks.
- `grid`: Generated metal2 followpins are snapped to the nearest routing track.
- `half_grid`: Generated metal2 followpins are snapped to the nearest half routing track.

route_special_core_pin_stop_route

```
route_special_core_pin_stop_route {atRowEnd atCellEnd}
```

Default: `atRowEnd`

Data_type: enum {RowEnd CellPinEnd}, read/write

Extends the follow pin wire either to the row-end or cell-end.

route_special_endcap_as_core

```
route_special_endcap_as_core {true | false}
```

Default: `false`

Data_type: bool, read/write

Specifies to treat Endcap macro as core macro. The option type is *bool*.

route_special_extend_nearest_target

```
route_special_extend_nearest_target {true | false}
```

Default: `false`

Data_type: bool, read/write

Extends the nearest target so that it can be connected. The option is of type *bool*.

route_special_jog_threshold_ratio

```
route_special_jog_threshold_ratio value
```

Default: `10`

Data_type: double, read/write

Defines the distance ratio between a jogging target and a straight target. The option is of type *float*.

route_special_layer_preferred_direction_cost

```
route_special_layer_preferred_direction_cost layer1 cost1 layer2 cost2...
```

Default: `""`

Data_type: string, read/write

Specifies layer cost of preferred direction. The option is of type `string`.

route_special_layer_non_preferred_direction_cost

```
route_special_layer_non_preferred_direction_cost layer1 cost1 layer2 cost2...
```

Default: `""`

Data_type: string, read/write

Specifies layer cost of non-preferred direction.

route_special_pad_pin_min_via_size

```
route_special_pad_pin_min_via_size viaSizePercent
```

Default: If you do not specify this parameter, the software uses a value of `20`, which permits vias that are 20% of the standard via size.

Data_type: double, read/write

Permits the software to reduce the size of vias at pad pins when changing layers, if there is not enough room for a standard size via.

route_special_pad_pin_split

```
route_special_pad_pin_split width spacing
```

Default: `""`

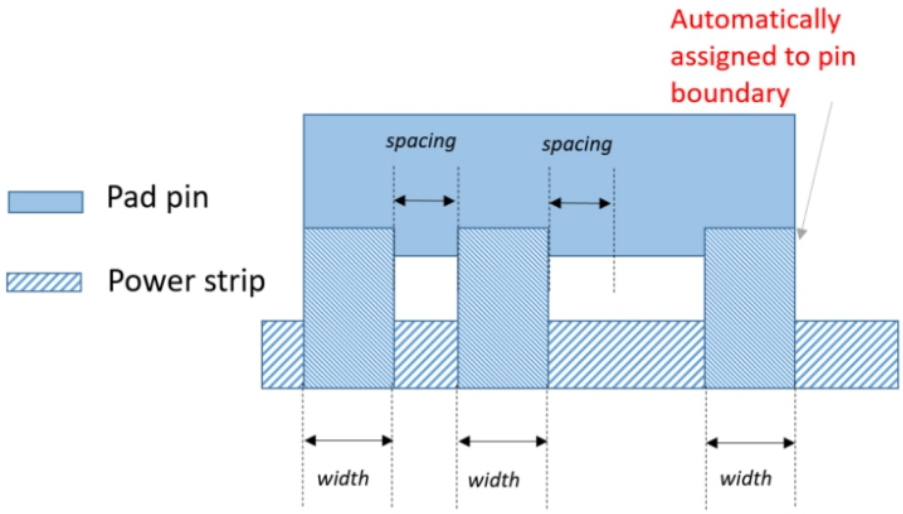
Data_type: string, read/write

Splits the width and spacing to connect wide pad pins.

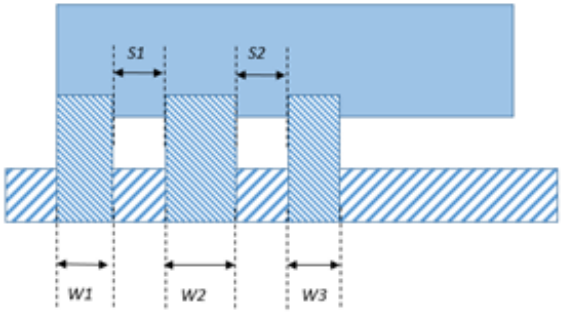
When you use {*width spacing*}, `route_special` repeats it to connect the wide pad pins. It stops when the rest distance cannot put a full width wire. When you use {*width1 spacing1 width2 spacing2 width3 spacing3 …*}, `route_special` sets the connection for wide pad pins only once. It stops when the rest distance cannot put a full width wire.

Refer to the command usage in various cases and corresponding outputs below:

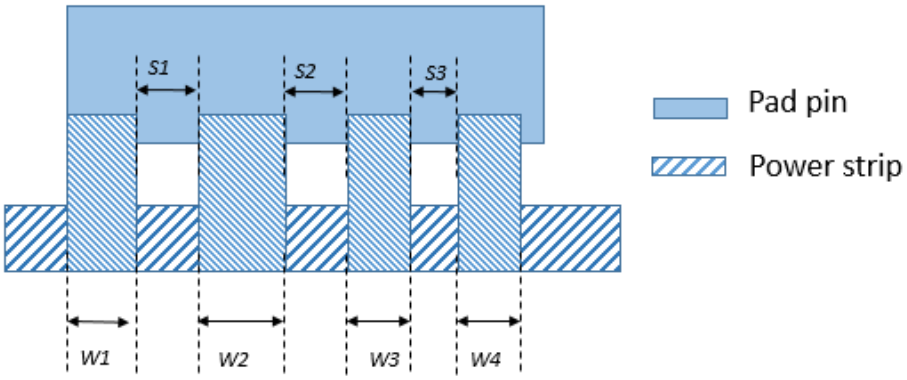
```
set_db route_special_pad_pin_split {width spacing}
```




```
set_db route_special_pad_pin_split {W1 S1 W2 S2 W3 S3}
```




```
set_db route_special_pad_pin_split {W1 S1 W2 S2 W3 S3 W4 S4 W5 S5 W6 S6}
```



route_special_pad_ring_use_lef

```
route_special_pad_ring_use_lef {true | false}
```

Default: `""`

Data_type: bool, read/write

Splits the width and spacing for wide pad pins.

route_special_pg_pin_as_signal

`route_special_pg_pin_as_signal {cellName: pinName}`

Default: `""`

Data_type: string, read/write

Lists the cell/pin information NOT extracted from CPF.

route_special_secondary_pin_max_gap

`route_special_secondary_pin_max_gap value`

Default: `0`

Data_type: double, read/write

Specifies variable value for gaps bigger than this value cause level shifter rail to break. The option type is *float*.

route_special_secondary_pin_rail_width

`route_special_secondary_pin_rail_width value`

Default: `0`

Data_type: double, read/write

Specifies width of followpin rail. The option type is *float*.

route_special_signal_pin_as_pg

`route_special_signal_pin_as_pg {true | false}`

Default: `false`

Data_type: bool, read/write

Specifies signal pins as PG pins in pad ring creation. When this attribute is set to `true`, the `route_special -nets` parameter supports wildcard characters (like * and ?).

route_special_split_long_via

`route_special_split_long_via {`*`threshold_value step_value offset_value height_value`*`}`

Default: If you do not specify this parameter, the software does not split vias.

Data_type: string, read/write

Splits vias longer than the specified length (`threshold_value`, in micrometers) into smaller vias with specified center-to-center spacing (`step_value`, in micrometers), left/bottom end offset (`offset_value`, in micrometers) and vertical/horizontal height (`height_value`, in micrometers) values. This is independent of the option `-split_via`.

Note: If the offset_value is given as negative, the vias will be placed symmetrically on both ends. The default value of `offset_value` is `-1`.

route_special_target_number

`route_special_target_number` *`value`*

Default: `0` (minimum). Maximum is `100000`.

Data_type: int, read/write

Specify number of target to connect. The option type is *int*.

route_special_target_search_distance

`route_special_target_search_distance` *`dist`*

Default: `0`

Data_type: double, read/write

Specifies target search distance. The option type is *float*.

route_special_time_limit

`route_special_time_limit` *`value`*

Default: `0`

Data_type: double, read/write

Specify time limit (in seconds) that the router is allowed in routing each port. The option type is *float*.

route_special_via_connect_to_shape

```
route_special_via_connect_to_shape {padring|ring|stripe|blockring|blockpin|coverpin|nos
hape|blockwire|corewire|followpin|iowire}
```

Default: padring ring stripe blockring blockpin coverpin noshape blockwire corewire followpin iowire

Data_type: enum* {padring ring stripe blockring blockpin coverpin noshape blockwire corewire followpin iowire}, read/write

Specifies which shapes vias can connect to.

The option type is *enum.*

route_special_via_through_to_closest_ring

```
route_special_via_through_to_closest_ring {true | false}
```

Default: If you do not specify this parameter, vias connect to all overlapping rings.

Data_type: bool, read/write

Specifies that a via should connect only to the closest ring layer when multiple rings overlap.

route_special_welltap_as_endcap

```
route_special_welltap_as_endcap string
```

Default: ""

Data_type: string, read/write

When used, `route_special` treats the `Welltap` cells as the `Endcap` cells. Additionally, the `ENDCAP` class cells are ignored by default, when connecting the `followpin` wires.

30

Prototyping Commands and Attributes

- FlexModel Commands and Attributes
- SoC Architecture Info (SAI) Commands

FlexModel Commands and Attributes

- assemble_proto_model
- convert_proto_model
- create_flexfiller_route_blockage
- create_proto_model
- create_proto_net_delay_model
- get_proto_model
- gui_highlight_proto_models
- identify_proto_model
- proto_design
- proto_design Category Attributes
- proto Category Attributes
- report_proto_models
- set_proto_model
- set_proto_model_physical_constraint
- set_proto_timing_settings
- write_repeater_rules

assemble_proto_model

```
assemble_proto_model
-top_dir {topDesignDir [protoModelDirName]}
[-block_dir {blockDesignDir [protoModeldirName]}]
-mmmc_file mmmcfileName
[-cpf_file cpfFileName]
[-help]
```

Generate full chip FlexModel netlist by assembling FlexModel partition netlists back to the top-level design for hierarchical FlexModel generation. It automatically creates a full chip FlexModel directory (proto_model) by merging back FlexModel information of all specified partition blocks and assembling back into the top-level design.

The purpose of this command is to get a FlexModel netlist DB in chip-level by assembling the block and top data. This command may be used to help memory usage by not loading a full chip netlist and can be run from where you will run top-down hierarchical flow. This flow should be used if there are partition block netlists.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `assemble_proto_model` parameter. For detailed description of the command and all its parameters, use the man command: `man assemble_proto_model` .
`-block_dir {blockDesignDir [protoModeldirName]}`	

	Specifies the path to block-level design. This directory should contain full netlist and FlexModel netlist of the block-level design and the corresponding proto model directory. If you specify more than one block-level design, then you must specify the `-block_dir` option separately for each block-level design. *`protoModelDirName`:* Optional. Specifies the directory name where FlexModel models can be found. The default directory name is retrieved from the setting of the `proto_create_dir` attribute that is specified before running `assemble_proto_model`. If `proto_create_dir` information is not available then the default directory name will be `proto_model` at the current run directory. It is recommended to specify *`protoModelDirName`* specially for `-block_dir` parameter. *Data_type*: string, optional
`-cpf_file`	Specifies chip level CPF file. This option should be specified for a design with power domain. *Data_type*: string, optional
`-mmmc_file`	Specifies chip level MMMC file. *Data_type*: string, optional
`-top_dir {topDesignDir [`*`protoModelDirName`*`]}`	
	Specifies the path to the top-level design directory. This directory should contain top-level design netlist. If the design has Flexmodels then the top-level FlexModel netlist should also be found in this directory with the corresponding proto model directory. *`protoModelDirName`:* Optional. Specifies the directory name where FlexModel models can be found. The default directory name is retrieved from the setting of the `proto_create_dir` attribute that is specified before running `assemble_proto_model`. If `proto_create_dir` information is not available then the default directory name will be `proto_model` at the current run directory. *Data_type*: string, optional

Examples

- The following command merges all models of partition block designs arb and tdsp_core to the top-level design DTMF_CHIP that does not have any FlexModels.

```
assemble_proto_model -top_dir {PTN_dir/DTMF_CHIP} \
-block_dir { PTN_dir/arb/arb_modelgen.enc.dat PTN_dir/arb/proto_model } \
-block_dir { PTN_dir/tdsp_core/tdsp_core_modelgen.enc.dat
PTN_dir/tdsp_core/proto_model} \
-mmmc_file viewDefinition.tcl
```

- The following command merges model of partition block ptn_wrapper to the top-level design dtmf_recvr_core that has FlexModel(s)

```
assemble_proto_model -top_dir {PTN/dtmf_recvr_core/modelgen.enc.dat proto_model} \
-block_dir { PTN/ptn_wrapper/modelgen.enc.dat PTN/ptn_wrapper/proto_model } \
-mmmc_file DBS/dtmf_recvr_core.enc.dat/viewDefinition.tcl
```

Related Information

- `create_proto_model`
- `get_proto_model`
- `proto` Category Attributes
- Prototyping Methodologies chapter in the *Innovus Stylus Common UI User Guide*

convert_proto_model

```
convert_proto_model
[-help]
[-out_dir string]
[-partition_dir string]
[-module_model_tag string -full_module_model_tag string]
```

Converts flexmodel netlist back to the original full netlist with changes at the partition block level whatever feedthrough insertion step is needed in the flow. You can use this command after using the `write_partitions` command. This command saves a directory under each partition directory with the default name *<designName>_proto_full.enc.dat* and you need to use this DB for block implementation.

To use this command in a FlexModel based partition flow:

1. Import a full chip FlexModel netlist design.
2. Floorplan the design.
3. Run the following commands to analyze the floorplan.
    a. `place_design`
    b. `route_early_global`
    c. `time_design -proto`
    d. `read_timing_debug_report -proto`
4. Do feedthrough insertion .
5. Do pin assignment
6. Optionally optimize the design.
7. Budget the timing for blocks.
8. Partition the design using the `commit_partitions` command.
9. Save the partition using `write_partitions -dir $ptnDir` (for placement and pin locations copy)
10. `convert_proto_model -partition_dir $ptnDir -out_dir $ptnDir`

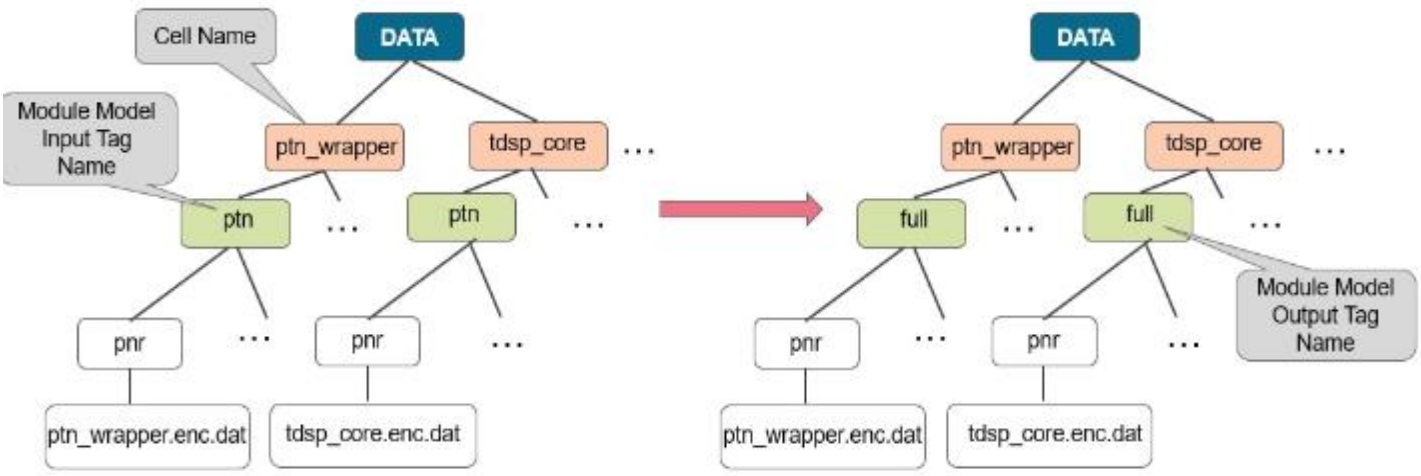
Parameters

`-help`	Outputs a brief description that includes type and default information for each `convert_proto_model` parameter. For a detailed description of the command and all of its parameters, use the man command: `man convert_proto_model`.
`-full_module_model_tag` *string*	Specifies the module model tag for the original full netlist that is used in the iHDB directory structure for saving the full hierarchical DB (*output*) after eco. **Note:** The `-full_module_model_tag` and `-module_model_tag` parameters are part of the iHDB flow and are used by the `convert_proto_model` command to convert the FlexModel netlist back to the original full netlist in the iHDB directory structure. Before using these parameters, you must set the default central directory name (`set_module_model -default_dir` *default_model_data_directory*) where the module models data will be saved/loaded. *Data_type*: string, optional
`-module_model_tag` *string*	Specifies the module model tag for the hierarchical DB (*input*). This module model tag is used in the Integrated Hierarchical Database flow. **Note:** The `-full_module_model_tag` and `-module_model_tag` parameters are part of the iHDB flow and are used by the `convert_proto_model` command to convert the FlexModel netlist back to the original full netlist in the iHDB directory structure. Before using these parameters, you must set the default central directory name (`set_module_model -default_dir` *default_model_data_directory*) where the module models data will be saved/loaded. *Data_type*: string, optional
`-out_dir` *string*	Specifies the directory to save post ECO full top/block netlist partition data. *Data_type*: string, optional
`-partition_dir` *string*	Specifies the partition directory for original full netlist. *Data_type*: string, optional

Example

The following command will read the FlexModel data with tag name “ptn” as input, and dump the original full netlist with tag name “full” in the iHDB directory structure.

```
convert_proto_model -module_model_tag ptn -full_module_model_tag full
```



Related Information

- `assemble_proto_model`
- `create_proto_model`
- `get_proto_model`
- `identify_proto_model`
- `report_proto_models`
- `set_proto_model`
- Prototyping Methodologies chapter in the *Innovus Stylus Common UI User Guide*

create_flexfiller_route_blockage

```
create_flexfiller_route_blockage
[-help]
```

Generates default congestion model to estimate routing blockages over FlexFillers based on user-specified percentage value for characterizing. This command can handle both module-based and instance group-based FlexModels. Routing blockage value of an instance group-based FlexModels will be derived based on its parent module routing blockage value. You should run this command when models are already created.

Note: This command should be used for ART based model generation flow. For non ART based flow, congestion model can still be created during `create_proto_model` command.

Note: The FlexFillers generation capability supports hybrid rows and generates FlexFillers that can fit in the floorplan rows and can be placed legally in the floorplan.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `create_flexfiller_route_blockage` parameter. For a detailed description of the command and all its parameters, use the man command: `man create_flexfiller_route_blockage`.

Examples

- The following example indicates that 10% of created models will be used for routing blockage characterization:

  ```
  create_flexfiller_route_blockage
  ```

- The following example indicates that 80% of created flexmodels will be used for routing blockage characterization and `tdsp_core` and `alu` modules should be part of these 80% models:

  ```
  set_db proto_include_model_route_blockage {tdsp_core alu}
  set_db proto_characterize_percentage_route_blockage 0.8
  create_flexfiller_route_blockage
  ```

- ```
  Begin pin_density
  # model_name pin_density flag
  # If flag is 1, use placeDesign and earlyGlobalRoute result for pin density.
  # If flat is 0, tool auto calculate it based on the pin area and total instance area.
  MOUT_DT_CTL 1.952371 1
  ```
  ```

```
ZRAS_RZ 2.361781 0
End pin_density
```

Related Information

- create_proto_model
- Prototyping Methodologies chapter in the *Innovus Stylus Common UI User Guide*

create_proto_model

```
create_proto_model
[-help]
[-keep_inst_file string]
[ [-out_dir string] [-module_model_tag string] ]
```

Creates prototyping models for the modules or instance groups that are marked as flexModels on a disk. By default, ART based flexModels are generated.

Note: To improve the usability, the `create_proto_model` command checks at the beginning if the current loaded design has any marking flexModels. If the design has marked some modules as flexModels, then the `create_proto_model` creates prototyping models for the modules or instance groups that are marked as flexModels. If no flexModel is marked, then it stops and displays a warning that no flexModel will be created. Alternatively, if the design has existing flexModels, which means the netlist has been reduced, then the `create_proto_model` command cannot be run and you need to switch to full netlist to re-generate it.

To generate prototyping models in a specific directory, type the following command:

```
set_db proto_create_dir directory_name
```

The default directory name is `proto_model`.

Note: This command supports multi-CPU settings and distributed processing.

```
set_multi_cpu_usage -remote_host 2 -cpu_per_remote_host 1
set_distributed_hosts -local
create_proto_model -out_dir
```

Note: This command can handle only upto 1000 flexModels in a design.

Note: Scan chains are supported during FlexModel creation.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `create_proto_model` parameter. For detailed description of the command and all its parameters, use the `man` command: `man create_proto_model`.
`-keep_inst_file` *string*	Specifies the name of the file that contains the list of instances to be kept specifically for FlexModel generation. **Note**: One instance name must be specified per line. *Data_type*: string, optional

`-module_model_tag` *string*	Specifies the module model tag to be used for ART based FlexModel generation in the iHDB directory structure. This module model tag is used in the Integrated Hierarchical Database flow.
	
	This parameter is part of the iHDB flow. Before using this parameter, you must set the default central directory name (`set_module_model -default_dir` *default_model_data_directory*) where the module models data will be saved/loaded.
	
	For example:
	
	If there is a design `ptn_wrapper` and the default central directory is `/project/DATA`
	on specifying
	`create_proto_model -module_model_tag proto_model`
	
	a flexModel will be created in the following directory:
	`/project/DATA/ptn_wrapper/proto_model/pnr`
	
	Note: The structure followed here is:
	
	<module_model_data_dir>/<design_name>/<tag_name>/<type>
	
	Data_type: string, optional

`-out_dir` *string*	Specifies a directory name where ART based FlexModel design will be saved. *Default:* `DBS/model_gen.enc` *Data_type*: string, optional

Examples

- The following command creates FlexModels for all the identified modules in the `proto_dir` directory:

```
set_db proto_identify_min_inst 2000
set_db proto_identify_max_inst 5000
identify_flexmodel
set_db proto_create_dir proto_dir
create_proto_model -out_dir DBS/model_design.enc
```

- The following command generates models using ART technique and save committed FlexModel design into DBS/model_design.enc.dat directory

```
create_proto_model -out_dir DBS/model_design.enc
```

Related Information

- `assemble_proto_model`
- `convert_proto_model`
- `get_proto_model`
- `identify_proto_model`
- `report_proto_models`
- `set_proto_model`
- Prototyping Methodologies chapter in the *Innovus Stylus Common UI User Guide*

create_proto_net_delay_model

```
create_proto_net_delay_model
[-help]
[-overwrite]
```

Derives psPM.model as a separate command. This command is used for active logic flat top model generation. For non-ART model generation, psPM.model can still be generated during `create_proto_model` step. It supports MMMC where it can automatically generate a psPM model file for each delay corner. with this, you can do corner dependent psPM budgeting and generated hold budgeting that is accurate where correct corner is used for hold instead of using one corner for both hold and setup.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `create_proto_net_delay_model` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man create_proto_net_delay_model`.
`-overwrite`	Overwrites an existing psPM.model. *Data_type*: bool, optional

Examples

The following commands generate psPM.model before running the `create_proto_model` command:

```
read_db DBS/init_design.enc.dat dtmf_recvr_core
set_db create_proto_net_delay_model
identify_proto_model
create_proto_model -flat_top DBS/model_gen.enc
```

Related Information

- `create_proto_model`
- `identify_proto_model`
- Prototyping Methodologies chapter in the *Innovus Stylus Common UI User Guide*

get_proto_model

```
get_proto_model
[-help]
[-committed]
[-create_opt_effort]
[-to_file string]
[-flip_table]
[-include_default]
[-tcl]
[-exclude_options string]
{-all | -model string | {[-type_match {flex_module flex_instgroup}]
 [-source_match {user auto}] }}
[-name | -type  | -create_total_area  | -create_gate_area  |
 -create_gate_count  | -create_area_per_gate  | -create_extra_macro |
 -plan_design_target_density | -flexfiller_route_blockage ]
```

Retrieves all the information that you specify on a model in a design. You can retrieve information on all the models or based on your search criteria and also on a specific module name. This commands returns the following information about `set_proto_model` parameters in the Innovus console:

- Parameter name
- Current value
- Type (Boolean, string, and so on)
- Whether the current value was set by user

If you do not specify a parameter, the Innovus software returns values for all of the `set_proto_model` parameters.

Parameters

`-help`	Prints a brief description that includes the type and default information for each `get_proto_model` parameter. For a detailed description of the command and all of its parameters, use the man command: `man get_proto_model`
`-all`	Reports the attributes of all modules. *Data_type*: bool, optional

`-committed`	Reports attributes of committed models only. *Data_type*: bool, optional
`-create_area_per_gate`	Reports the specified area per gate value. *Data_type*: bool, optional
`-create_total_area`	Reports the total area including the standard cell area and the macro area. *Data_type*: bool, optional
`-create_extra_macro`	Reports the specified additional macros. *Data_type*: bool, optional
`-create_gate_area`	Reports the specified gate area value. *Data_type*: bool, optional
`-create_gate_count`	Reports the specified gate count value. *Data_type*: bool, optional
`-create_opt_effort`	Reports the specified optimization setting. *Data_type*: bool, optional
`-exclude_options` *string*	Excludes specified options. *Data_type*: string, optional
`-flexfiller_route_blockage`	Reports percentages of flexFiller route blockage. *Data_type*: bool, optional
`-flip_table`	Flips orientation of the table. *Data_type*: bool, optional
`-include_default`	Includes public options with default parameters in parentheses (for non-tcl output). *Data_type*: bool, optional
`-model` *string*	Reports the attributes of specified models. *Data_type*: string, optional

`-name`	Reports the module name. **Note:** Does not support wildcards. *Data_type*: bool, optional
`-plan_design_target_density` *number*	
	Specifies the target utilization value to derive initial module size of a model. *Data_type*: bool, optional
`-source_match {user auto}`	Reports the source match. Valid values are `user` and `auto`. *Default*: user auto *Data_type*: (enum)+, optional
`-tcl`	Reports the model as Tcl list. *Default*: table print *Data_type*: bool, optional
`-to_file` *string*	Allows you to output results to a file. *Data_type*: string, optional
`-type`	Reports the module type. *Data_type*: bool, optional
`-type_match {flex_module flex_instgroup}`	
	Reports the type match. Valid values are: `flexmodel` and `full`. *Default*: flex_module *Data_type*: (enum)+, optional

Examples

- The following `get_proto_model` command retrieves the type, source, total area, and the target utilization of the `alu` and `crossbar` modules.

```
set_proto_model -model alu -type flex_module -create_total_area 1000 \
 -plan_design_target_density 0.75
set_proto_model -model crossbar -type full -create_total_area 3000
get_proto_model -all
Model    type        source  create_total_area planDesign_target_util
alu      flex_module  user       1000              0.75
crossbar full         user       3000
```

- The following command returns the values of all models in the Tcl list format.

```
get_proto_model -all  -tcl
{{name alu}  {type flex_module} {source_match user} {create_total_area 1000}
{plan_design_target_density 0.75}}
{{name crossbar}  {type full} {source_match user} {create_total_area 3000}}
```

- The following command only returns FlexModels information in Tcl list format

```
get_proto_model -type_match {flex_module}  -tcl
{{name alu}  {type flex_module} {source_match user} {create_total_area 1000}
{plan_design_target_density 0.75}}
```

Related Information

- assemble_proto_model
- create_proto_model
- convert_proto_model
- identify_proto_model
- report_proto_models
- set_proto_model
- Prototyping Methodologies chapter in the *Innovus Stylus Common UI User Guide*

gui_highlight_proto_models

```
gui_highlight_proto_models
[-help]
[-keep_existing_highlight]
[-types {flex_module flex_instgroup partition power_domain}]
[-level integer] | {[-min_level integer][-max_level integer]}
[-reset]
```

Highlights objects, such as, FlexModels, partitions, and/or power domains. This command automatically resets the list of used colors for each partition, so that the same color(s) can be reused in different partitions.

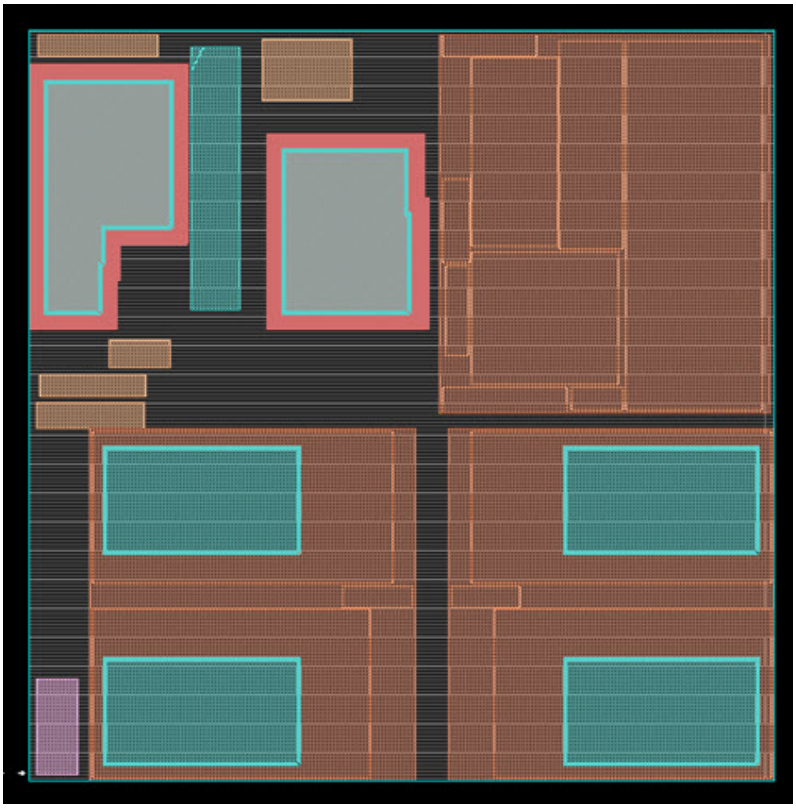
Parameters

`-help`	Prints out the command usage.
`-keep_existing_highlight`	Highlights new objects in addition to the existing highlighted objects. *Data_type*: bool, optional
`-level integer`	Specifies the level to highlight. *Data_type*: int, optional
`-max_level integer`	Highlights objects from level 0 to the specified level. *Data_type*: int, optional
`-min_level integer`	Minimum level to highlight. *Data_type*: int, optional
`-reset`	Resets all the previously highlighted models. *Data_type*: bool, optional
`-types {flex_module flex_instgroup partition power_domain}`	
	Allows you to specify which model type(s) to be highlighted. *Default*: `flex_module flex_instgroup` *Data_type*: (enum)+, optional

Examples

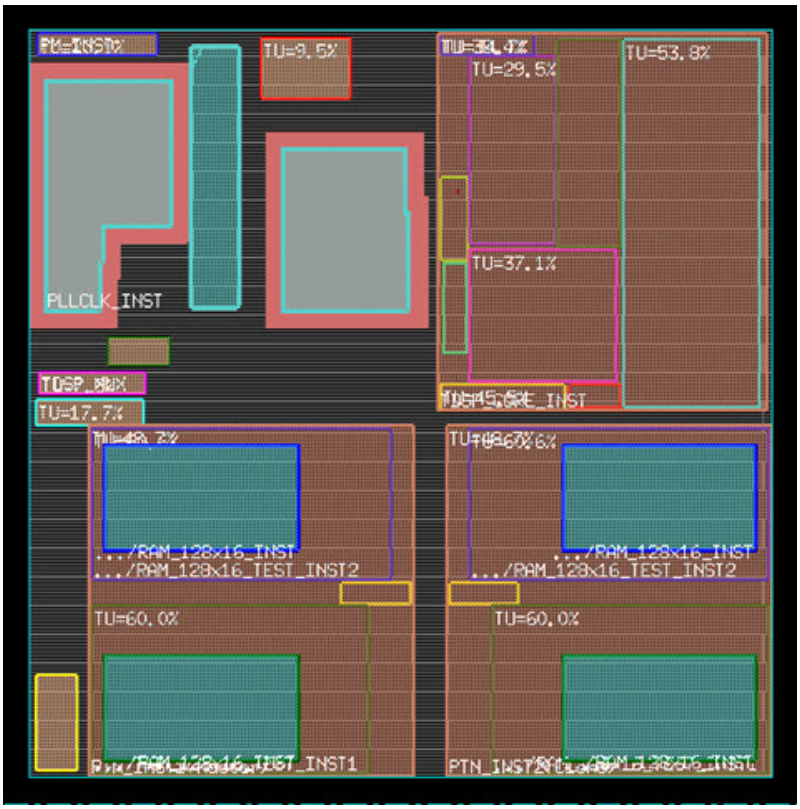
The example below has the level range from 0 to 1. So we have `min_level` 0 and `max_level` 1. Default `gui_highlight_proto_models` works levels from min to max.

Before `gui_highlight_proto_models`



`gui_highlight_proto_models` or
`gui_highlight_proto_models -min_level 0 -max_level 1` or
`gui_highlight_proto_models -min_level 0 #`

These three usages have the same result on this design.



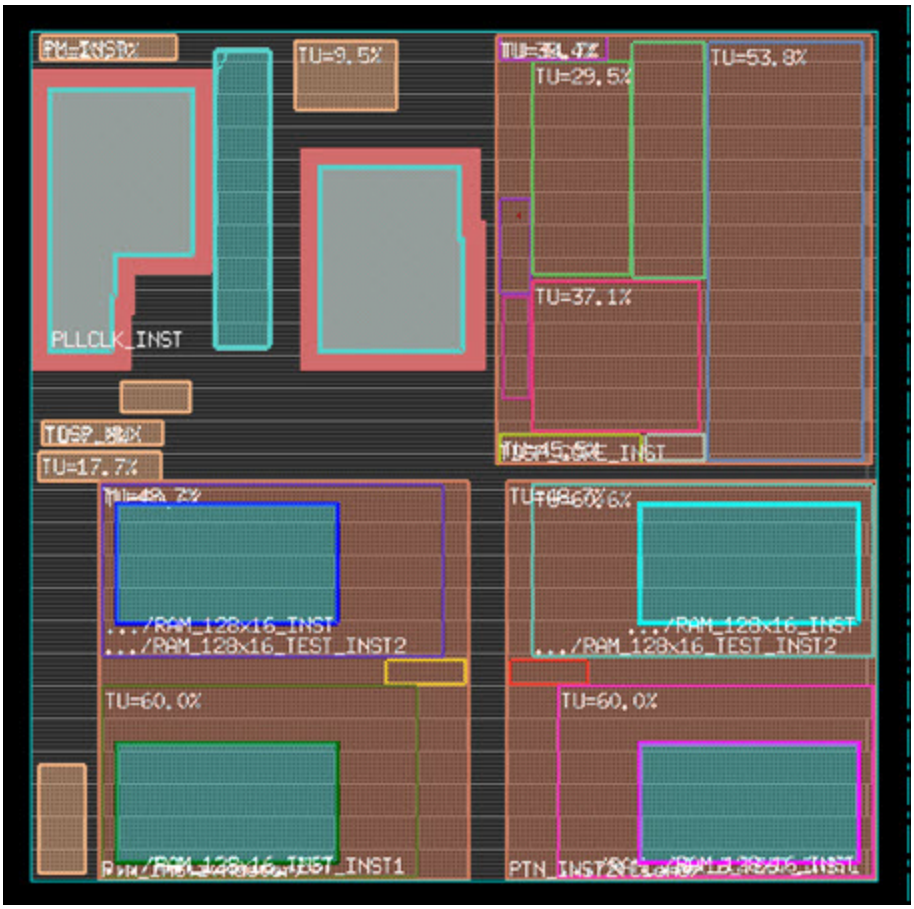

```
gui_highlight_proto_models -level 1 #
```

Only display flexModels on the specified level.

Or

```
gui_highlight_proto_models -min_level 1 #
```

Display flexModels upper or equal with level 1.



Related Information

- `set_proto_model`
- Prototyping Methodologies chapter in the *Innovus Stylus Common UI User Guide*

identify_proto_model

```
identify_proto_model
[-help]
[-skip_report]
```

Identifies the modules or instance groups of a design that will become FlexModels. This command supports either module-based or instance group-based FlexModels. If a design netlist has no logical hierarchy, the command will automatically create instance groups based on placement clustering algorithm and identify them as FlexModels.

The `identify_proto_model` command uses the following proto Category Attributes:

```
set_db proto_identify_min_inst number
set_db proto_identify_max_inst number
proto_identify_exclude_module moduleName
proto_identify_exclude_module_tree moduleTree
```

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `identify_proto_model` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man identify_proto_model`.
`-skip_report`	Skips reporting the list of identified flexModels at the end of the command. *Data_type*: bool, optional

Examples

- Consider the following design hierarchy that has 130000 instances at the top level. `MOD1` has 60000 instances, `MOD2` has 40000 instances and `MOD3` has 30000 instances.

```
TOP (130000)
          MOD1 (60000)
                MOD1a (20000)
                MOD1b (10000)
                MOD1c (30000)
          MOD2 (40000)
                MOD2a (20000)
                MOD2b (20000)
          MOD3 (30000)
                MOD3a (20000)
```

- The following commands identify the modules with minimum 20000 instances and maximum 50000 instances, and also excludes the module `MOD3` from being selected as flexModel.

```
set_db proto_identify_min_inst 20000
set_db proto_identify_max_inst 50000
set_db proto_identify_exclude_module {MOD3}
identify_proto_model
get_proto_model -type flex_module -tcl -name MOD1c MOD2 MOD3a
```

  In this example, MOD1a, MOD1c, MOD2, and MOD3a are being marked as flexModels. Here, MOD1 does not get marked as a flexModel because it is more than the maximum instance threshold. MOD1b is less than minimum instance threshold so it is also not marked as a flexModel. MOD3 is excluded because we want to exclude it as mentioned above. However its sub-module MOD3a is marked as a flexModel because it satisfies the minimum instance threshold.

Related Information

- assemble_proto_model
- create_proto_model
- convert_proto_model
- get_proto_model
- report_proto_models
- set_proto_model
- Prototyping Methodologies chapter in the *Innovus Stylus Common UI User Guide*

proto_design

```
proto_design
[-help]
[-constraint_file name]
[-summary file_name]
```

Generates an initial floorplan for prototyping that can be used as a start point for making the final floorplan. It is a super command that places flexModels, power domains, and/or user-specified module seeds.

Note: The `proto_design` command supports multiple levels of hierarchy, such as nested partitions and nested power domains. To support this feature, you can use the `multi_level` option of the `proto_design_level` attribute in the proto Category Attributes. You also have a choice to run proto_design to place Flexmodels and/or macro blocks for each level of hierarchy for better placement QOR.

By default, the `proto_design` command makes settings internally so that FlexModel floorplans are available even though any user-defined settings are not made. However, if you make any changes, then the `proto_design` command settings will be overwritten. This feature helps to improve the usability.

Note: The `proto_design` command honors the fence spacing rules.

Parameters

`-constraint_file` *name*	Specifies the constraint file name (string). Uses the constraints specified in the constraint file for fence creation and module guide generation. A constraint file is a text file that can contain a list of seeds used when generating fences and module guides, and basic relative, spacing and orientation constraints to follow during macro placement. *Data_type*: string, optional
`-help`	Outputs a brief description that includes type and default information for each `proto_design` parameter. For a detailed description of the command and all of its parameters, use the man command: `man proto_design`
`-summary` *file_name*	Specifies the name of the file in which the summary output report with some checkPoints like violation summary, TU of modules, master/clones shape are written. *Data_type*: string, optional

Examples

The following example invokes proto_design with the specified constraint file `proto_user_constraint.txt` file with all the default settings, and saves the design floorplan of the iteration that gives the best result:

```
set_db proto_design_write_floorplan best
proto_design -constraint_file proto_user_constraint.txt
```

Related Information

- `proto_design` Category Attributes
- Prototyping Methodologies chapter in the *Innovus Stylus Common UI User Guide*

proto_design Category Attributes

The root attributes in the `proto_design` category are:

- proto_design_cover_fixed_macros
- proto_design_effort
- proto_design_flexmodel_constraint_type
- proto_design_keep_guide
- proto_design_place_macro
- proto_design_remove_overlap

To set the value of any root attribute, use the `set_db` command:

```
set_db attribute_name value
```

To get the current value of any root attribute, use the `get_db` command:

```
get_db attribute_name
```

To get the current value of all attributes in the `proto_design` category, use the following command:

```
get_db -category proto_design
```

proto_design_cover_fixed_macros

`proto_design_cover_fixed_macros {true | false}`

Default: `false`

Data_type: bool, read/write

When enabled it makes the `proto_design` command aware of congestion.

Related Commands

- `proto_design`

proto_design_effort

`proto_design_effort {low|high}`

Default: `low`

Data_type: enum={low high}, read/write

Specifies the effort level used by the `proto_design` command.

Related Commands

- `proto_design`

proto_design_flexmodel_constraint_type

`proto_design_flexmodel_constraint_type {guide|region|fence}`

Default: `region`

Data_type: enum={guide region fence}, read/write

Specifies the type of constraint the FlexModel is converted to after `proto_design`.

Related Commands

- `proto_design`

proto_design_keep_guide

`proto_design_keep_guide {true|false}`

Default: `true`

Data_type: bool, read/write

Retains module guides after the floorplan has been created.

Related Commands

- `proto_design`

proto_design_place_macro

`proto_design_place_macro {true|false}`

Default: `true`

Data_type: bool, read/write

Specifies if macro placement is done when using the `proto_design` command.

Related Commands

- `proto_design`

proto_design_remove_overlap

```
proto_design_remove_overlap {true|false}
```

Default: `true`

Data_type: bool, read/write

Specifies if overlap should be removed when using the `proto_design` command.

Related Commands

- `proto_design`

proto Category Attributes

The root attributes in the `proto` category are:

- proto_allow_model_with_io
- proto_allow_power_domain_in_flexmodel
- proto_create_dir
- proto_create_high_fanout_ps_per_micron
- proto_create_lib
- proto_create_metal_fill_ndr
- proto_create_metal_fill_nominal
- proto_create_multi_corner_ps_per_micron
- proto_create_ndr_ps_per_micron
- proto_create_no_flex_filler
- proto_create_partition_as_flexmodel
- proto_create_pipeline_flop
- proto_create_power_domain_ps_per_micron
- proto_create_timing_budget
- proto_identify_engine
- proto_identify_estimated_flexmodel_number
- proto_identify_exclude_module
- proto_identify_exclude_module_and_parent
- proto_identify_exclude_module_tree
- proto_identify_honor_objects_hierarchy
- proto_identify_max_inst
- proto_identify_min_inst
- proto_keep_inst_file_only
- proto_keep_instance_defined_in_sdc
- proto_keep_slack_improve_ndr

- proto_max_report_ndr_net
- proto_route_net_ndr
- proto_timing_net_delay
- proto_use_timing_net_delay_model
- proto_verbose

To set the value of any root attribute, use the `set_db` command:

```
set_db attribute_name value
```

To get the current value of any root attribute, use the `get_db` command:

```
get_db attribute_name
```

To get the current value of all attributes in the `proto` category, use the following command:

```
get_db -category proto
```

proto_allow_model_with_io

`proto_allow_model_with_io {true | false}`

Default: `false`

Data_type: bool

Allows modules with I/O to be selected as FlexModel.

Related Commands

- `place_design`
- `time_design`
- `identify_proto_model`
- `create_proto_model`

proto_allow_power_domain_in_flexmodel

`proto_allow_power_domain_in_flexmodel {true|false}`

Default: `false`

Data_type: bool

Allows modules that contains power domain(s) to be selected as FlexModel.

Related Commands

- `place_design`
- `time_design`
- `identify_proto_model`
- `create_proto_model`

proto_create_dir

`proto_create_dir` *`directoryName`*

Default: `proto_model`

Data_type: string

Specifies the directory to save created models.

Related Commands

- `place_design`
- `time_design`
- `create_proto_model`

proto_create_high_fanout_ps_per_micron

`proto_create_high_fanout_ps_per_micron {true | false}`

Default: `true`

Data_type: bool

Create high fanout psPM models.

Related Commands

- place_design
- time_design
- create_proto_model

proto_create_lib

`proto_create_lib` *`libraryName`*

Default: `proto_model_lib`

Data_type: string

Specifies the OpenAccess library to save created models.

Related Commands

- place_design
- time_design
- create_proto_model

proto_create_metal_fill_ndr

`proto_create_metal_fill_ndr` *`double`*

Default: 0.0

Data_type: float

Creates metal fill that is used by `create_ps_per_micron_model` for creating psPM model for nets with non default rules.

Related Commands

- place_design
- time_design
- create_proto_model

proto_create_metal_fill_nominal

`proto_create_metal_fill_nominal` *double*

Default: 0.5

Data_type: float

Creates metal fill that is used by create_ps_per_micron_model for creating psPM model for normal nets.

Related Commands

- `place_design`
- `time_design`
- `create_proto_model`

proto_create_multi_corner_ps_per_micron

`proto_create_multi_corner_ps_per_micron {true | false}`

Default: `false`

Data_type: bool

Create multi corner psPM models.

Related Commands

- `place_design`
- `time_design`
- `create_proto_model`

proto_create_ndr_ps_per_micron

`proto_create_ndr_ps_per_micron {auto | on | off}`

Default: `auto`

Data_type: enum {auto on off}

Specifies whether `create_ps_per_micron_model` should generate psPM model for non default rule or not

Related Commands

- `place_design`
- `time_design`
- `create_proto_model`

proto_create_no_flex_filler

```
proto_create_no_flex_filler {true|false}
```

Default: `false`

Data_type: bool

Specifies whether to create proto model without adding flex filler.

Related Commands

- `place_design`
- `time_design`
- `create_proto_model`

proto_create_partition_as_flexmodel

```
proto_create_partition_as_flexmodel {true| false}
```

Default: `false`

Data_type: bool

Creates partition specified as flexmodel automatically.

Related Commands

- place_design
- time_design
- create_proto_model

proto_create_pipeline_flop

```
proto_create_pipeline_flop {*AB*}
```

Default: {}

Data_type: string

Specifies a list of pipeline flop instance patterns that are used in the design. The default value is an empty list.

Related Commands

- place_design
- time_design
- create_proto_model

proto_create_power_domain_ps_per_micron

```
proto_create_power_domain_ps_per_micron {true | false}
```

Default: false

Data_type: bool

Creates power domain psPM models.

Related Commands

- place_design
- time_design
- create_proto_model

proto_create_timing_budget

```
proto_create_timing_budget {true | false}
```

Default: `false`

Data_type: bool

When enabled, it uses Active-Logic Reduction Technology (ART) in timing budget mode during FlexModel generation. The FlexModel creation uses the same settings as the `create_timing_budget` command to preserve logic paths so that they yield good timing budgets later on in the flow.

Note: This option should only be specified if design has partitions defined.

Related Commands

- `create_timing_budget`
- `time_design`
- `create_proto_model`

proto_identify_engine

```
proto_identify_engine {auto|module_based|instgroup_based}
```

Default: auto

Data_type: enum {auto module_based instgroup_based}

Identifies the proto algorithm. Specifying `instgroup_based` enables instance group based `identify_proto_model`.

Related Commands

- `place_design`
- `time_design`
- `create_proto_model`

proto_identify_estimated_flexmodel_number

`proto_identify_estimated_flexmodel_number` *number*

Default: 20

Data_type: int

Specifies the estimated flexModel number to identify for the design when using instance group based `identify_proto_model`.

Related Commands

- `place_design`
- `time_design`
- `create_proto_model`

proto_identify_exclude_module

`proto_identify_exclude_module {`*module1 module2* `...}`

Default: { }

Data_type: string

Excludes the specified module(s) but not its sub-modules for being marked as models.

Related Commands

- `place_design`
- `time_design`
- `identify_proto_model`

proto_identify_exclude_module_and_parent

`proto_identify_exclude_module_and_parent {`*module1 module2* `...}`

Default: { }

Data_type: string

Excludes specified module(s) and their parents for being marked as models.

Related Commands

- `place_design`
- `time_design`
- `identify_proto_model`

proto_identify_exclude_module_tree

`proto_identify_exclude_module_tree { module1 module2 ...}`

Default: { }

Data_type: string

Excludes the specified module(s) and its sub-modules for being marked as flexModels.

Related Commands

- `place_design`
- `time_design`
- `identify_proto_model`

proto_identify_honor_objects_hierarchy

`proto_identify_honor_objects_hierarchy {object_list}`

Default: { }

Data_type: string

Specifies the hierarchy of the list of objects that the instance group based `identify_proto_model` will honor.

Related Commands

- place_design
- identify_proto_model
- time_design

proto_identify_max_inst

```
proto_identify_max_inst number
```

Default: -1

Data_type: int

Specifies the maximum instance count threshold of a module to be marked as model. The default value is eight percent of the total instance count and goes no lower than 300000 instances.

Related Commands

- place_design
- time_design
- identify_proto_model

proto_identify_min_inst

```
proto_identify_min_inst number
```

Default: 1000

Data_type: int

Specifies the minimum instance count threshold of a module that is to be marked as a model.

Related Commands

- place_design
- time_design
- identify_proto_model

proto_keep_inst_file_only

```
proto_keep_inst_file_only {true | false}
```

Default: `false`

Data_type: bool

Specifies that the instance list for flexmodels should only be kept in the `keep_inst_file`.

Related Commands

- `place_design`
- `time_design`
- `identify_proto_model`
- `create_proto_model`

proto_keep_instance_defined_in_sdc

```
proto_keep_instance_defined_in_sdc {true | false}
```

Default: `false`

Data_type: bool

Specifies that the flexmodel will keep the instance defined in the sdc file.

Related Commands

- `place_design`
- `time_design`
- `identify_proto_model`
- `create_proto_model`

proto_keep_slack_improve_ndr

```
proto_keep_slack_improve_ndr {true | false}
```

Default: `false`

Data_type: bool

Specifies that report_NDR_WNS_gain will keep top critical nets as non default rule nets.

Related Commands

- place_design
- time_design
- identify_proto_model
- create_proto_model

proto_max_report_ndr_net

```
proto_max_report_ndr_net double
```

Default: 0.05

Data_type: float

Specifies that create_ps_per_micron_model creates psPM model for nets with non default rules.

Related Commands

- place_design
- time_design
- identify_proto_model
- create_proto_model

proto_route_net_ndr

```
proto_route_net_ndr ruleName
```

Default: { }

Data_type: string

Creates psPM model for nets with non default rules.

Related Commands

- place_design
- time_design
- identify_proto_model
- create_proto_model

proto_timing_net_delay

`proto_timing_net_delay {M` *number1* `[:M` *number2* `]` *number3* `...}`

Default: {}

Data_type: string

Specifies the estimated delay per micron for each routing layer (DB or LEF layer names). You can also specify layer pair value for each view.

Related Commands

- place_design
- time_design
- identify_proto_model
- create_proto_model

proto_use_timing_net_delay_model

`proto_use_timing_net_delay_model {best_layer_no_detour|best_layer_blockage_aware|use_actual_wire}`

Default: `best_layer_no_detour`

Data_type: enum {best_layer_no_detour best_layer_blockage_aware use_actual_wire}

Allows you to the set mode of calculating wire delay.

Note: Using the `best_layer_blockage_aware` option, Innovus uses an obstacle aware approach for

estimating the channel-routing-wire-length between two timing points. The `time_design -prototype` command is aware of routing blockages and other partition fences when the `best_layer_blockage_aware` option is specified.

Related Commands

- `place_design`
- `time_design`
- `identify_proto_model`
- `create_proto_model`

proto_verbose

`proto_verbose`

Default: false

Data_type: bool

Outputs detailed information into the log file.

Related Commands

- `place_design`
- `time_design`

report_proto_models

```
report_proto_models
[-help]
[-constraint]
[-out_file string]
[-min_inst integer |-identify |-create]
[-min_inst integer]
[-instance_ratio float | -min_inst integer]
```

Reports statistics information for models.

Note: The `report_proto_models` command takes the top-level cells in account while reporting the netlist reduction ratio at the end.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `report_proto_models` parameter. For a detailed description of the command and all of its parameters, use the man command: `man report_proto_models`
`-constraint`	Report physical constraint information. *Data_type*: bool, optional
`-create`	Reports the statistics, including reduction ratio of the models that have been created. *Data_type*: bool, optional
`-identify`	Reports the statistics of the modules/instgroups that have been identified as models. *Data_type*: bool, optional
`-instance_ratio` *float*	Specifies ratio value where `report_proto_models` only reports flexModels that have instance ratio greater than or equal to this specified value. *Min:* 0 *Max:* 1.79769313486232e+308 *Data_type*: float, optional
`-min_inst` *integer*	Specifies the threshold of an instance count that is used to filter modules. *Data_type*: int, optional
`-out_file` *string*	Specifies output report file name. *Data_type*: string, optional

Examples

- The following command reports modules or instance groups have been identified as FlexModels

```
report_proto_models -identified
```

```
-------------------------------------------------------------------------------------------------------
                  Model Instance Summary
-------------------------------------------------------------------------------------------------------
Mod  Mod    Source  #Insts
Num  Type           Stdcell  Cell/GroupName        HinstName
-------------------------------------------------------------------------------------------------------
  1  FlexMod  auto       106 CPF_ISO_MOD_1         CPF_ISO_HIER_INST_107
  2  FlexMod  auto        33 dma                   DMA_INST
  3  FlexMod  user        52 power_manager         PM_INST
  4  FlexMod  user        33 ram_128x16_test       PTN_INST1/RAM_128x16_TEST_INST1
  5  FlexMod  user        33 ram_128x16_test       PTN_INST1/RAM_128x16_TEST_INST2
  6  FlexMod  user        33 ram_128x16_test       PTN_INST2/RAM_128x16_TEST_INST1
  7  FlexMod  user        33 ram_128x16_test       PTN_INST2/RAM_128x16_TEST_INST2
  8  FlexMod  user        33 tdsp_data_mux         PTN_INST1/TDSP_MUX
  9  FlexMod  user        33 tdsp_data_mux         PTN_INST2/TDSP_MUX
 10  FlexMod  user        33 tdsp_data_mux         TDSP_MUX
 11  FlexMod  user        34 spi                   SPI_INST
 12  FlexMod  auto       560 alu_32                TDSP_CORE_INST/ALU_32_INST
 13  FlexMod  auto        59 CPF_LS_MOD_4889       TDSP_CORE_INST/CPF_LS_HIER_INST_61
 14  FlexMod  auto        49 data_bus_mach         TDSP_CORE_INST/DATA_BUS_MACH_INST
 15  FlexMod  auto       107 decode_i              TDSP_CORE_INST/DECODE_INST
 16  FlexMod  auto       828 execute_i             TDSP_CORE_INST/EXECUTE_INST
 17  FlexMod  user        37 mult_32               TDSP_CORE_INST/MPY_32_INST
 18  FlexMod  user        44 port_bus_mach         TDSP_CORE_INST/PORT_BUS_MACH_INST
 19  FlexMod  user        42 prog_bus_mach         TDSP_CORE_INST/PROG_BUS_MACH_INST
 20  FlexMod  auto       650 tdsp_core_glue        TDSP_CORE_INST/TDSP_CORE_GLUE_INST
 21  FlexMod  auto        76 ulaw_lin_conv         ULAW_LIN_CONV_INST
-------------------------------------------------------------------------------------------------------
           2908 out of 3332 (87.3% of chip's stdcells are in models)
Ending "report_proto_model" (total cpu=0:00:00.0, real=0:00:00.0, peak res=349.9M, current mem=400.7M)
```

- The following command reports models in the design with hierarchical modules that have at least 30 instances:

```
report_proto_models -min_inst 30
```

```
          #STDCELLS          #MACROS
    Total  Parent Children   Parent Children  HinstName                                  CellName
-------------------------------------------------------------------------------------------------------
M     106     106        0        0        0  CPF_ISO_HIER_INST_107                      CPF_ISO_MOD_1
M      76      76        0        0        0  ULAW_LIN_CONV_INST                         ulaw_lin_conv
M      33      33        0        0        0  TDSP_MUX                                   tdsp_data_mux
      2640     211     2429        0        0  TDSP_CORE_INST                             tdsp_core
M      59      59        0        0        0  TDSP_CORE_INST/CPF_LS_HIER_INST_61         CPF_LS_MOD_4889
       33      23       10        0        0  TDSP_CORE_INST/TDSP_CORE_MACH_INST         tdsp_core_mach
M     650     427      223        0        0  TDSP_CORE_INST/TDSP_CORE_GLUE_INST         tdsp_core_glue
M      42      23       19        0        0  TDSP_CORE_INST/PROG_BUS_MACH_INST          prog_bus_mach
M      44      24       20        0        0  TDSP_CORE_INST/PORT_BUS_MACH_INST          port_bus_mach
M      37       0       37        0        0  TDSP_CORE_INST/MPY_32_INST                 mult_32
M     828     673      155        0        0  TDSP_CORE_INST/EXECUTE_INST                execute_i
M     107      66       41        0        0  TDSP_CORE_INST/DECODE_INST                 decode_i
M      49      26       23        0        0  TDSP_CORE_INST/DATA_BUS_MACH_INST          data_bus_mach
M     560     277      283        0        0  TDSP_CORE_INST/ALU_32_INST                 alu_32
M      34      29        5        0        0  SPI_INST                                   spi
      124      27       97        0        2  PTN_INST2                                  ptn_wrapper
M      33      33        0        0        0  PTN_INST2/TDSP_MUX                         tdsp_data_mux
M      32       0       32        1        0  PTN_INST2/RAM_128x16_TEST_INST2            ram_128x16_test
M      32       0       32        1        0  PTN_INST2/RAM_128x16_TEST_INST1            ram_128x16_test
      124      27       97        0        2  PTN_INST1                                  ptn_wrapper
M      33      33        0        0        0  PTN_INST1/TDSP_MUX                         tdsp_data_mux
M      32       0       32        1        0  PTN_INST1/RAM_128x16_TEST_INST2            ram_128x16_test
M      32       0       32        1        0  PTN_INST1/RAM_128x16_TEST_INST1            ram_128x16_test
M      52      43        9        0        0  PM_INST                                    power_manager
M      33      32        1        0        0  DMA_INST                                   dma
-------------------------------------------------------------------------------------------------------
      3325       3     3322        3        4  dtmf_recvr_core                          dtmf_recvr_core
-------------------------------------------------------------------------------------------------------
    2904                            4          MODEL TOTALS     dtmf_recvr_core
Total Num of Models = 16
Ending "report_proto_model" (total cpu=0:00:00.2, real=0:00:00.0, peak res=349.9M, current mem=400.8M)
^
```

- The following command reports all models after the FlexModels have been committed:

  `report_proto_models -create`

```
=============================================================================================================
                                        commit_proto_model Report
=============================================================================================================
Mod      #Instances        #Instances                #
Num   Stdcell   Filler   MODEL   Actual    Ratio   Macro   CellName/GroupName   HinstName
=============================================================================================================
  1       107        0     107      107    1.000       0   CPF_ISO_MOD_1        CPF_ISO_HIER_INST_107
  2        59        0      59       59    1.000       0   CPF_LS_MOD_4889      TDSP_CORE_INST/CPF_LS_HIER_INST_61
  3       555        0     555      555    1.000       0   alu_32               TDSP_CORE_INST/ALU_32_INST
  4        30        0      30       41    0.732       0   data_bus_mach        TDSP_CORE_INST/DATA_BUS_MACH_INST
  5        77        1      78      137    0.569       0   decode_i             TDSP_CORE_INST/DECODE_INST
  6        23        1      24       67    0.358       0   dma                  DMA_INST
  7       472        8     480      838    0.573       0   execute_i            TDSP_CORE_INST/EXECUTE_INST
  8         0       37      37        0      N/A       0   mult_32              TDSP_CORE_INST/MPY_32_INST
  9        26        0      26       33    0.788       0   port_bus_mach        TDSP_CORE_INST/PORT_BUS_MACH_INST
 10        50        0      50       69    0.725       0   power_manager        PM_INST
 11        25        0      25       31    0.806       0   prog_bus_mach        TDSP_CORE_INST/PROG_BUS_MACH_INST
 12        32        0      32       33    0.970       1   ram_128x16_test      PTN_INST1/RAM_128x16_TEST_INST1
 13        32        0      32       33    0.970       1   ram_128x16_test      PTN_INST1/RAM_128x16_TEST_INST2
 14        32        0      32       33    0.970       1   ram_128x16_test      PTN_INST2/RAM_128x16_TEST_INST1
 15        32        0      32       33    0.970       1   ram_128x16_test      PTN_INST2/RAM_128x16_TEST_INST2
 16        29        3      32       62    0.516       0   spi                  SPI_INST
 17       749        0     749      749    1.000       0   tdsp_core_glue       TDSP_CORE_INST/TDSP_CORE_GLUE_INST
 18        20        0      20       20    1.000       0   tdsp_data_mux        PTN_INST1/TDSP_MUX
 19        20        0      20       20    1.000       0   tdsp_data_mux        PTN_INST2/TDSP_MUX
 20        20        0      20       20    1.000       0   tdsp_data_mux        TDSP_MUX
 21        91        0      91       91    1.000       0   ulaw_lin_conv        ULAW_LIN_CONV_INST
=============================================================================================================
        2481       50    2531     3031    0.835       4   SUBTOTAL
                                     320                0   top-level stdcells
=============================================================================================================
        2801       50    2851     3351    0.851       4   TOTAL
=============================================================================================================
Ending "report_proto_model" (total cpu=0:00:00.2, real=0:00:02.0, peak res=256.0M, current mem=542.6M)
```

- If `-constraint` option is specified, the report has the has *PhysicalConstraint* column

  `report_proto_models -create`

```
================================================================================
                          create_proto_model Report
================================================================================
Mod      #Instances        #Instances                    #
Num   Stdcell   Filler   MODEL   Actual     Ratio   Macro   NrRef   CellName/GroupName
================================================================================
  1         0       37      37        0     0.000       0       1   mult_32
  2        26        0      26       33     0.788       0       1   port_bus_mach
  3        50        0      50       69     0.725       0       1   power_manager
  4        25        0      25       31     0.806       0       1   prog_bus_mach
  5        32        0      32       33     0.970       1       4   ram_128x16_test
  6        29        3      32       62     0.516       0       1   spi
  7        20        0      20       20     1.000       0       3   tdsp_data_mux
================================================================================
          318       40     358      387     0.925       4
================================================================================
```

```
report_proto_models -create -constraint
```

```
================================================================================================================
                                        create_proto_model Report
================================================================================================================
Mod      #Instances        #Instances                    #
Num   Stdcell   Filler   MODEL   Actual     Ratio   Macro   NrRef   CellName/GroupName   PhysicalConstraint
================================================================================================================
  1         0       37      37        0     0.000       0       1   mult_32              region {370.4 235.03 451.4 453.91}
  2        26        0      26       33     0.788       0       1   port_bus_mach        region {338.0 235.03 368.6 248.71}
  3        50        0      50       69     0.725       0       1   power_manager        region {21.405 445.36 92.405 457.33}
  4        25        0      25       31     0.806       0       1   prog_bus_mach        region {263.6 267.52 277.4 320.53}
  5        32        0      32       33     0.970       1       4   ram_128x16_test      fence {53.8 17.86 218.2 117.04}
  6        29        3      32       62     0.516       0       1   spi                  guide {20.6 19.57 44.642 76.0}
  7        20        0      20       20     1.000       0       3   tdsp_data_mux        region {202.8 118.75 243.6 130.72}
================================================================================================================
          318       40     358      387     0.925       4
================================================================================================================
```

Related Information

- assemble_proto_model
- create_proto_model
- convert_proto_model
- get_proto_model
- identify_proto_model
- set_proto_model
- Prototyping Methodologies chapter in the *Innovus Stylus Common UI User Guide*

set_proto_model

```
set_proto_model
[-help]
-model name_list
[-type {full | flex_module | flex_inst_group}]
[[-create_total_area number ]| [-create_gate_area number ]|
 [{-create_gate_count number -create_area_per_gate number}]]
[-create_extra_macros {macro_name number [macro_name number] ...}
[-flexfiller_route_blockage {Mnumber1[:Mnumber2] percentage …}]
[-plan_design_target_density number]
[-reset]
```

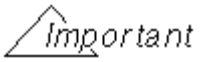
The `set_proto_model` command specifies the model type of a specific module.

Note: Wildcards are not supported by this command.

Parameters

`-create_area_per_gate` *number*	
	Specifies the area per gate. *Data_type*: float, optional
`-create_extra_macros {macro_name` *number*`[macro_name` *number*`] ...}`	
	Specifies the additional macro(s) of a module that you want to add in the original netlist. If you wish to add extra macros, this option changes the netlist. For example, if your original netlist has macro `A` and you specify an additional macro named macro `B`, then the flexModel with have both macro `A` and macro `B`. *Data_type*: string, optional
`-create_gate_area` *number*	Specifies the total area of gates. This value does not include macro area. Its unit is square microns. The total area (equivalent to `-create_total_area`) is calculated as the sum of specified total gate area and macro area. *Data_type*: float, optional

`-create_gate_count` *number*	Specifies the number of gates. If you specify this parameter, you also need to specify area per gate value.Total gate area is equal to the product of gate count and area per gate. The calculated total gate area does not include the macro area. So, the total area (equivalent to `-create_total_area`) is equal to the sum of total gate area and the macro area. *Data_type*: int, optional
`-create_total_area` *number*	Specifies the module area of a model that includes the standard cell area and the macro area. **Note**: A warning message is issued when macro area is greater than the area specified using the `-create_total_area` parameter. **Note:** For `-create_area_per_gate number`, `-create_total_area number`, `-create_gate_area number` and `-create_gate_count number`; you can define the area instead of the value calculated by the tool (for times when you wish to specify larger area for the module to achieve better results). *Data_type*: float, optional
`-help`	Outputs a brief description that includes type and default information for each `set_proto_model` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man set_proto_model`.
`-model` *name_list*	Specifies a name for the model type. *Data_type*: string, required
`-plan_design_target_density` *number*	

	Specifies the target utilization value to derive initial module size of a model. This value is used by `proto_design` to derive the module size of a specified flexModel if its module is not placed yet. *Default*: current design utilization **Note:** You can reduce the target utilization value by 10 to 20 percent to account for 10 to 20 percent increase in instance count during optimization.  If the post-target utilization is 0.75, then Cadence recommends you to set the following: `set_proto_model -model` *`name_list`* `-plan_design_target_density 0.55` *Data_type*: float, optional
`-reset`	Resets all the attributes of a module. *Data_type*: bool, optional
`-type {full | flex_module | flex_inst_group}`	
	Specifies the model type as full or a flex module or a flexInstgroup. You use this parameter when you want to convert a flex_module back to the original netlist or converse. *Data_type*: enum, optional

Examples

- The following command specifies the number of gates as 10000 and area per gate as 0.65 for the tdsp_core model:

```
set_proto_model -model tdsp_core -create_gate_count 10000 \
-create_area_per_gate 0.65 -type flex_module
```

- The following command specifies model alu as flex_module with total area `1000` and target utilization `0.75`:

```
set_proto_model -model alu -type flex_module -create_total_area 1000 \
-plan_design_target_util 0.75
```

- The following command specifies model crossbar as flex_module with total area `3000`:

```
set_proto_model -model crossbar -type flex_module -create_total_area 3000
```

- The following command specifies the extra macro `{pllclk 2 ram128x164}` of the `chico` model:

```
set_proto_model -model chico -create_extra_macros {pllclk 2 ram128x164}
```

- The following command resets all the attributes of the `alu` model:

```
set_proto_model -model alu -reset
```

Related Information

- `assemble_proto_model`
- `create_proto_model`
- `convert_proto_model`
- `get_proto_model`
- `identify_proto_model`
- `report_proto_models`
- Prototyping Methodologies chapter in the *Innovus Stylus Common UI User Guide*

set_proto_model_physical_constraint

```
set_proto_model_physical_constraint
[-help]
-type {guide | fence | region | soft_guide | none}
[-models string]
```

Sets flexModels to specific physical constraints to guide placement. By default, all the flexModels will be set to the specified type.

Parameters

`-help`	Prints out the command usage.
`-models` *string*	Specifies a list of flexmodels. *Data_type*: string, optional
`-type {guide | fence | region | soft_guide | none}`	
	Specifies the constraint type. You can select one of the following options: • `guide` • `fence` • `region` • `soft_guide` • `none` *Data_type*: enum, required

Examples

The following commands change physical constraint of all flexmodels to guide before running detailed placement:

```
set_proto_model_physical_constraint -type guide
place_opt_design
route_early_global
time_design -proto
read_timing_debug_report -proto
```

Related Information

- `set_proto_model`
- Prototyping Methodologies chapter in the *Innovus Stylus Common UI User Guide*

set_proto_timing_settings

```
  set_proto_timing_settings
[-help]
[-reset]
```

Sets or resets the prototyping settings in the flow. The following variables are set by the command:

- `Input Pin Transition`
  (*Default: 0.1* ps)
- `Net Load`
  (*Default:* 0.5pF)
- `Net Delay`
  (*Default: 1000ps*)
- `Using Default Delay Limit`
  (*Default: 1000*)
- `set_db delaycal_enable_high_fanout true`
  (*Default:* false)

Parameters

`-help`	Outputs a brief description that includes type and default information for each `set_proto_timing_settings` parameter. For a detailed description of the command and all of its parameters, use the man command: `man set_proto_timing_settings`.
`-reset`	Resets all prototyping settings. The settings are enabled by default. It resets the variables mentioned above and rebuilds the timing graph. It also resets the following setting: `set_db delaycal_ignore_net_load 0` The above is set to true by `time_design -prototype`, and must be reset. *Data_type*: bool, optional

Related Information

- `time_design`
- Prototyping Methodologies chapter in the *Innovus Stylus Common UI User Guide*

write_repeater_rules

```
write_repeater_rules
[-help]
{ [-buffers listOfBufferNames] | [-inverters listOfInverterNames] }
[-out_file string]
[-layer listOfLayerNames]
-max_target_slew float
-min_target_slew float
[-slew_interval float]
[-report string]
```

Automatically calculates how much distance a specified list of buffers or inverters can drive across a set of routing layers and generates a rule file for the repeaters

For each repeater and routing layer a two pin net is created with given repeater as source and sink. Based on timing, the distance between source and sink is automatically adjusted until slew at the sink is close to the same as the input slew at the source. The command outputs a repeater rule file with one entry for each specified repeater with the distance value that gives the least total delay among all slews and layers. This repeater rule file can be used as an input to the repeater-by-rule command: `add_repeater_by_rule`.

Here is a sample rule file:

```
SetBufferDrivingStrength bufferName maxDistanceInMicrons maxCapLoad
SetBufferDrivingStrength ...
SetDefaultDrivingStrength maxDistanceInMicrons maxCapLoad
SetDistLastRepToSink maxDistanceInMicrons
```

There is more information on repeater rule syntax here: `add_repeater_by_rule`

Parameters

`-help`	Prints out the command usage
`-buffers listOfBufferNames`	List of buffers. Buffers should have only one input and one output. *Data_type*: string, optional
`-inverters listOfInverterNames`	List of inverters. Inverters should have only one input and one output. *Data_type*: string, optional

`-layer` *listOfLayerNames*	Specifies the layer names. *Default*: All available routing layers. *Data_type*: string, optional
`-max_target_slew` *float*	Maximum target slew in picoseconds. *Data_type*: float, required
`-min_target_slew` *float*	Minimum target slew in picoseconds. *Data_type*: float, required
`-out_file` *string*	Specifies the repeater rule file name. The rule is created in a sub-directory named *proto_model* under the current directory. *Default*: `<designName>_repeater.rule.` *Data_type*: string, optional
`-report` *string*	Specifies the report file name. *Default* : `<designName> _buffer_psPM.rpt.` *Data_type*: string, optional
`-slew_interval` *float*	Calculates the step and slew range interval. Specified value should be within [0.01 1.0]. The step is calculated using the following: `step = ($max_target_slew - $min_target_slew) * $slew_interval` *Default*: 0.1 *Data_type*: float, optional

Examples

- The following command reports distances and pico-second delay values per micron for specified buffers `buf2` with given minimum and maximum slews:

  ```
  write_repeater_rules -buffers {buf2} -max_target_slew 100 -min_target_slew 70
  ```

- The following command reports distance and pico-second delay values per micron for specified buffers TBUFX1, TBUFX3, and TBUFX20 with given minimum, maximum slews, and

slew interval:

```
write_repeater_rules -buffers {TBUFX1 TBUFX3 TBUFX20} -min_target_slew 90 -
max_target_slew 120 -slew_interval 0.8
```

- The following command reports distances and pico-second delay values per micron for specified inverter CLKINVX8 with given minimum, maximum slews, and slew interval:

```
write_repeater_rules -inverters CLKINVX8 -min_target_slew 70.0 -max_target_slew 80.0
-slew_interval 1.0
```

Related Topics

- `add_repeater_by_rule`
- Prototyping Methodologies chapter in the *Innovus Stylus Common UI User Guide*

SoC Architecture Info (SAI) Commands

- add_sai_boundary_flops
- end_sai
- read_sai
- report_sai_constraint

add_sai_boundary_flops

```
add_sai_boundary_flops
[-help]
[-clock string]
[-delete_flops]
-hinsts string
-ports string
[-ref_flop string [-clock_input_pin string]
[-data_input_pin string]
[-data_output_pin string]]
```

Allows insertion of boundary flops for empty modules/blocks. The command options specify the clock net name, the module name and the module pin name.

Parameters

`-clock` *`string`*	Specifies clock net name. *Data_type*: string, optional
`-clock_input_pin` *`string`*	Specifies the cell CLK port name. *Data_type*: string, optional
`-data_input_pin` *`string`*	Specifies the cell D port name. *Data_type*: string, optional
`-data_output_pin` *`string`*	Specifies the cell Q port name. *Data_type*: string, optional
`-delete_flops`	Removes all the flops that have been added. *Data_type*: string, optional
`-help`	Outputs a brief description that includes type and default information for each `add_sai_boundary_flops` parameter.
`-hinsts` *`string`*	Specifies the module name. **Note:** While specifying multiple hinsts and multiple ports, ensure that the number of specified hinsts is the same as the number of specified ports. *Data_type*: string, optional
`-ports` *`string`*	Specifies the module pin name. *Data_type*: string, optional
`-ref_flop` *`string`*	Specifies the flop cell name to be inserted along the boundary. *Data_type*: string, optional

Related Information

- `read_sai`
- `end_sai`
- Prototyping Methodologies chapter in the *Innovus Stylus Common UI User Guide*
- Syntax and Scripts chapter in the *Innovus Stylus Common UI User Guide*
    - Supported SAI Commands

end_sai

```
end_sai
[-help]
```

Disables the SoC Architecture Information (SAI) mode.

Parameters

`-help`	Prints the command usage.

Related Information

- `read_sai`
- Prototyping Methodologies chapter in the *Innovus Stylus Common UI User Guide*
- Syntax and Scripts chapter in the *Innovus Stylus Common UI User Guide*
  - Supported SAI Commands

read_sai

```
read_sai
[-help]
[name]
[-partial]
[-reduce_by_flexfiller]
[-rule_only]
```

Enables the SoC Architecture Information (SAI) mode that allows the use of SAI commands in batch mode.

Parameters

`-help`	Prints out the command usage.
name	Specifies the SAI file name. *Data_type*: string, optional
`-partial`	When specified, it enables SAI to support designs loaded using the `read_db` command. Use this option to enable partial netlist support in the SAI mode. *Data_type*: bool, optional
`-reduce_by_flexfiller`	When specified, it reduces the size of the generated SAI netlist using the FlexFiller technology that is same as the FlexModel methodology. This helps to address the capability and run-time limitations of a growing design size (10 or 100 of million instances). *Data_type*: bool, optional
`-rule_only`	When specified, the rules are imported only in the SAI file. *Data_type*: bool, optional

Example

The following command reads in the *example2.0.sai* file and reduces the SAI netlist while maintaining the module areas:

```
read_sai example2.0.sai -reduce_by_flexfiller
```

Related Information

- `end_sai`
- Prototyping Methodologies chapter in the *Innovus Stylus Common UI User Guide*
- Syntax and Scripts chapter in the *Innovus Stylus Common UI User Guide*
  - Supported SAI Commands

report_sai_constraint

```
report_sai_constraint
[-help]
[-area_weight float]
[-spacing_weight float]
[-order_weight float]
[-congestion_weight float]
[-rule {all | rule_name | rule_name_list}]
```

You can use the `report_sai_constraint` command to validate a floorplan result and to check whether the specified SAI constraints are met. This enables front-end designers to easily evaluate a floorplan provided by their physical designers against the specified floorplan rules without requiring them to have much knowledge about using the Innovus System. The `report_sai_constraint` command reports:

- The general floorplan violations such as macro-to-macro overlap, fence-to-fence overlap, macro overlaps with fence, and/or if macros are placed outside their parent module boundary.
- SAI 2.0 constraint violations
- The Floorplan Quality Index that enables you to estimate whether a floorplan is ready to be used. Floorplan Quality Index is similar to the total negative slack (TNS) concept of timing analysis
- Support area, spacing, net weight, order, and congestion quality indexes; and their index weights.

Besides generating a report, it also creates a violation marker if violations are detected. These

violation markers will be displayed in the Innovus artwork window and can be queried from the Innovus violation browser.

Based on the reported floorplan quality index, you can decide whether this floorplan is good to go or not. If the floorplan quality index does not meet your requirements, you can further:

- Adjust the wire length, congestion, and/or overlap cost factors and rerun the proto_design command to produce new floorplan result(s).
- Improve the current floorplan by manual adjustment.

Note: Use this command in the SAI interactive mode or write in the SAI file.

Parameters

`-area_weight` *float*	Specifies the weight of the area index. *Default*: 1.0 *Data_type*: float, optional
`-congestion_weight` *float*	Specifies the weight of the congestion index. *Default*: 1.0 *Data_type*: float, optional
`-help`	Outputs a brief description that includes type and default information for each `report_sai_constraint` parameter.
`-order_weight` *float*	Specifies the weight of the order index. *Default*: 1.0 *Data_type*: float, optional
`-rule {all |` *rule_name* `|` *rule_name_list*`}`	
	Checks and reports the violation(s) of the specified rule(s). *Default*: all (All SAI rules of the current design) *Data_type*: string, optional
`-spacing_weight` *float*	Specifies the weight of the area index. *Default*: 1.0 *Data_type*: float, optional

Related Information

- `read_sai`
- `end_sai`
- Prototyping Methodologies chapter in the *Innovus Stylus Common UI User Guide*
- Syntax and Scripts chapter in the *Innovus Stylus Common UI User Guide*
    - Supported SAI Commands

31

Rail Analysis Commands

- add_what_if_shapes
- connect_die_package
- convert_stream_to_def
- generate_esd_rlrp_report
- map_dies
- report_differential_voltage
- report_irdrop_regions
- report_noise_margin
- extract_package
- report_esd_network
- report_esd_voltage
- report_joule_heat
- report_package
- report_rail
- report_resistance
- report_self_heat
- report_signal_resistance
- scale_what_if_capacitance
- scale_what_if_current
- scale_what_if_resistance
- set_advanced_package_options
- set_advanced_rail_options
- set_die_model
- set_dynamic_rail_simulation
- set_multi_die_analysis_mode
- set_net_group
- set_off_chip_package_trace
- set_package
- set_pg_nets
- set_power_data
- set_power_pads
- set_rail_analysis_config
- set_rail_analysis_domain
- set_self_heat_analysis_mode
- set_voltage_regulator_module
- view_package_results
- write_current_region
- write_die_model
- write_hier_view

- write_power_pads
- write_power_up_report

add_what_if_shapes

```
add_what_if_shapes
[-help]
[-area {x1 y1 x2 y2}]
[-display]
[-direction {hor | ver}]
[-offset {X Y}]
[-add -method {auto | exact}]
[-layer layerName]
[-nets {net1 net2 net3…}]
[-spacing value]
[-objects {selected | all} ]
[-add  | -select | -deselect |
 -delete | -list  | -write filename | -read filename]
[-pitch pitch]
[-add -width width]
[-type {wire | via}]
[-format {txt def}]
[-write filename -append ]
[-remove_existing_wires]
```

Specifies to add, select, deselect, delete, list, and display the what if wires/vias to a design.

Parameters

`-help`	
	Outputs the command usage.
`-add`	
	Specifies to add what-if wires/vias. *Data_type*: bool, optional
`-append`	
	Specifies to append all what-if wires/vias to the existing file specified with the `-save filename` parameter. *Data_type*: bool, optional
`-area {x1 y1 x2 y2}`	
	Specifies the rectangular region in which the what if wires/vias are to be generated. *Data_type*: rect, optional
`-delete`	
	Specifies to delete all or selected what-if shapes by constraint `-net/-layer/-type`. You can use this parameter in conjunction with the `-objects` parameter. *Data_type*: bool, optional
`-deselect`	
	Specifies to deselect all what-if shapes by constraints `-net/-layer/-type`. Use this parameter in conjunction with `-area` to deselect what-if shapes overlapping with the specified area. *Data_type*: bool, optional

`-direction {hor | ver}`

Controls the direction of the wires added.

Data_type: enum, optional

`-display`

Invokes the GUI to show what-if shapes and zooms to the selected shapes. It works only in the `-win` mode.

Data_type: bool, optional

`-format {txt def}`

Specifies the output file format. The `def` format specifies to create incremental DEF to load into GUI using `read_def -top`. The default output format is `txt`.

Data_type: enum, optional

`-layer` *layerName*

Specifies the LEF layer name for wires, or layer pair for via to be added.

Data_type: string, optional

`-list`

Specifies to list all or selected what-if shapes by constraint `-net/-layer/-type` . You can use this parameter in conjunction with the `-objects` parameter.

Data_type: bool, optional

`-method {auto | exact}`

Specifies the method used to add vias:

- `auto`: When area and layer pair are specified, the tool automatically adds vias on all intersecting points on the specified layer pairs in the given area.
- `exact`: When an area and layer pair are specified, the tool adds a via of exact size of the specified area between the specified layer pair.

Data_type: enum, optional

`-nets {`*net1 net2 net3…*`}`

Specifies the name of the nets for which what-if wires/vias is to be created. The what-if engine will create a shape based on the net pattern.

Data_type: string, optional

`-objects {selected | all}`

Specify `-objects` in conjunction with the `-delete` or `-list` parameters to delete or list the selected what-if shapes. You can also delete or list all what-if shapes. The default value is `all`.

For example, you can use the following command to delete the selected objects:

```
add_what_if_shapes -delete -objects selected
```

Data_type: enum, optional

`-offset {X Y}`

Specifies to create metal/via from the area + offset. This parameter is useful when you run multiple what-if shape creation commands and want to keep the same area but have different offset.

Data_type: point, optional

`-pitch` *pitch*

Specifies the distance between the centre line of two adjacent what-if wires (pitch) to be drawn in the specified `-area` in the preferred direction of the specified `-layer`. The direction of created wires can also be controlled using the `-direction` parameter.

If this parameter is not provided, a single rectangular shape of size {llx,lly,urx,ury} is created.

Data_type: float, optional

`-read` *filename*

	Specifies to load the saved what-if shape file. *Data_type*: string, optional
`-remove_existing_wires`	
	Specifies to remove a wire and connected vias from the specified layer and area before adding what-if shapes. *Data_type*: bool, optional
`-select`	
	Specifies to select all what-if shapes by constrains `-net/-layer/-type`. Use this parameter in conjunction with `-area` to select what-if shapes overlapping with the specified area. *Data_type*: bool, optional
`-spacing` *value*	
	Specify the spacing between each net. Use this parameter to control the spacing between each net. The `-pitch` value specified should be greater than number of nets * (width+spacing). *Data_type*: string, optional
`-type {wire | via}`	
	Specifies what kind of shape is to be created. *Data_type*: enum, optional
`-width` *value1 value2 ..valuen*	
	Specifies the width of what-if wires. You can use the `-width` parameter to specify a separate width value for each net. *Data_type*: string, optional
`-write` *filename*	

	Specifies to save all the in-memory what-if wires/vias to the specified file. The format of the file is:
	`type wire net netName area llx lly urx ury layer layer1`
	`type via net netName area llx lly urx ury layer layer1 layer2`
	Following is an example of the file containing one wire and one via shape:
	`type wire net VDD area 191.000000 763.000000 207.000000 772.000000 layer METTOP`
	`type via net VDD area 513.000000 497.700000 517.000000 499.100000 layer METG1 ME`
	Data_type: string, optional

Examples

- The following command adds a what-if wire for the net `VDD` and on the LEF layer `METTOP`:

```
add_what_if_shapes -type wire -net VDD -area {391.000000 663.000000 407.000000 672.000000} -layer METTOP -add
add_what_if_shapes -type wire -net VDD -area {191.000000 663.000000 207.000000 772.000000} -layer METTOP -add -pitch 9 -width 0.4
```

- The following command creates a what-if via for the net `VDD` and on the LEF layers `METG1 MET5`:

```
add_what_if_shapes -type via -net VDD -area {513.000000 497.700000 517.000000 499.100000} -layer {METG1 MET5} -add -method exact
add_what_if_shapes -type via -area {0 0 1000 1000} -method auto -layer {METTOP METG1} -add -net VDD
```

- The following command specifies to list selected what-if shapes:

```
add_what_if_shapes -list -objects selected
```

- The following command specifies to select and highlight all wires in the win mode:

```
add_what_if_shapes -select
```

- The following command specifies to select and highlight all what-if vias using area:

```
add_what_if_shapes -type via -area {500 490 530 500} -select -net VDD
```

- The following command specifies to save what-if shapes into the file `whatif.shapes`:

```
add_what_if_shapes -write whatif.shapes
```

- The following command specifies to delete all what-if shapes in memory:

```
add_what_if_shapes -delete -objects all
```

- In the following example, for nets "`VDD VDD1 VDD2 VSS`", three spacing is possible (between VDD and VDD1, between VDD1 and VDD2, and between VDD2 and VSS), that is, spacing is "0.5 0.6 0.7" and width of what-if wires for all nets is "0.5":

```
add_what_if_shapes -net {VDD VDD1 VDD2 VSS} -pitch 4 -width 0.5 -spacing {0.5 0.6 0.7} -area {0 0 12 12} -type wire -add -layer M10 -direction hor
```

- The following command will remove all wires on `M7` in the `0,0` to `100,100` region as well as any vias connected to the deleted wires:

```
add_what_if_shapes -remove_existing_wires -layer {M7} -area { 0 0 100 100 }
```

Related Information

- `scale_what_if_capacitance`
- `scale_what_if_resistance`
- `scale_what_if_current`
- "What-If Rail Analysis" in the *Voltus Stylus Common UI User Guide*

connect_die_package

```
connect_die_package
[-help]
[-output_dir directory]
-package_model_file filename
{-die_mcp_header_file filename | -net_pad_file_pair_list {{<net1> <pad file1>} {<net2> <pad file2>}...}}
```

The command launches the *MCP Editor* GUI to make manual connections between package and die pins. It also performs the following functions:

- maps the voltage source files with the package model to check and correct mapping information before rail analysis.
- generates a mapping file that can be used in previous versions of the software in case there is no mapping between die and package pins.

The command has two types of inputs:

- Voltage source files in the xy format and a package model file with the MCP header.
- A circuit file with the MCP header and a package model file with the MCP header.

The output will be the same for both types of inputs:

- A ckt mapping file containing only the MCP header, with the name `pkg_model_base_name + "_map"+suffix`.
- The file named '`un_mapped_bumps.txt`' will contain all the bumps which cannot be mapped.

Parameters

`-help`	Outputs the command usage and a brief description about the command parameters.
`-die_mcp_header_file filename`	
	Specifies a circuit file with the MCP header containing the die abstract. *Data_type*: string, optional
`-net_pad_file_pair_list {{<net1> <pad file1>} {<net2> <pad file2>}...}`	
	Specifies the voltage source files mapping to each net. *Data_type*: string, optional
`-output_dir directory`	
	Specifies the output directory containing the mapping results. *Data_type*: string, optional
`-package_model_file filename`	
	Specifies the name of the package model file. *Data_type*: string, required

Examples

- The following command specifies a circuit file with MCP header and a package model file with the MCP header:
  ```
  connect_die_package -die_mcp_header_file super_filter_mcp_ploc.ckt -package_model_file C1_Voltus_4L_C1_Voltus_4L_Final_PinBaseSPICE.ckt
  -output_dir out_ploc
  ```
- The following command specifies voltage source files in the XY format and a package model file with the MCP header:
  ```
  connect_die_package -output_dir ./out_pad -net_pad_file_pair_list {{VDD VDD_AO_srcs.pp} {VDD VDD_external_srcs.pp} {VSS VSS_srcs.pp}} -
  package_model_file C1_Voltus_4L_C1_Voltus_4L_Final_PinBaseSPICE.ckt
  ```

Related Information

- `set_rail_analysis_config`
- "Package Analysis" in the *Voltus Stylus Common UI User Guide*

convert_stream_to_def

```
convert_stream_to_def
[-help]
[-black_box_cell_file filename]
-stream_file filename
-stream_layer_map filename
[-inst_name_file filename]
-net_location_file filename
[-output_components {ALL | NONE | CONN}]
-output_def_file filename
[-output_lef_macros filename]
-power_grid_library libraryname
-top_cell cellname
[-use_stream_label {ALL | TOP | <tech_layer_name> | <cell_name>:<tech_layer_name> | <cell_name>:ALL}]
[-verilog_file filename]
[-spef_file filename]
[-esd_marker_layer_file filename]
[-esd_cell_bbox_file filename]
[-inst_name_prefix prefix]
[-ignore_stream_inst_name]
[-print_complete_short_path]
[-extract_signal_net_logic_info_only]
[-zero_pin_size]
```

A GDS2DEF utility that converts a given GDSII file to a DEF file for power/rail signoff analysis. The utility can convert both cell-level GDS and full-chip level GDS to DEF. The generated DEF contains all the geometries for the P/G routing, the specified P/G pin information, component and logical connectivity information.

Parameters

`-help`	Outputs the command usage.
`-black_box_cell_file filename`	

	Specifies the name of a file containing a list of black box cells that the utility is to ignore. You can specify both exact string matches and wildcard searches in the specified file.
	For exact string matches, each cell name must reside on a separate line in the file, as shown below:
	`CellA`
	`CellB`
	The wildcard search feature allows you to search cell names that contain wildcard characters, such as square brackets and underscore. When you want to perform wildcard search, type the keyword `wildcard` first, followed by the cell name. For example, if you want to ignore cell names which contains CLK, add one line `wildcard .*CLK.*`.
	Data_type: string, optional

`-esd_marker_layer_file filename`	

Specifies a file that includes a list of ESD cell/instance names and their corresponding GDS marker layers that represent the bounding box of the ESD cells. The format of the file is:
`<cell_name> <marker_layer> [top_layer] [bottom_layer]`

Here,

- `cell_name` is the ESD cell/instance name
- `marker_layer` could contain multiple GDS layers, separated by ";"
- The optional `top_layer` and `bottom_layer` specifies the top and bottom technology layers of the ESD cell pin shape. You can use these optional arguments to specify the top and bottom technology layers of the ESD cell pin shape. This provides you with the flexibility to specify the lowest layers to be used for creating pins in the LEF. In the following example, only the shapes between M3 and M1 layers are ESD cell pin shapes:

Marker file example:

```
ESD_diode 168;255:3 M3 M1
ESD_clamp 122 M3 M1
```

Data_type: string, optional

`-esd_cell_bbox_file` *filename*

Specifies a file that includes the bounding box for each ESD cell. The format of the file is:
`<cell_name> <x1 y1 x2 y2> [top_layer] [bottom_layer]`

Here,

- `cell_name` is the ESD cell/instance name
- `x1 y1 x2 y2` are the bounding box coordinates
- The optional `top_layer` and `bottom_layer` specifies the top and bottom technology layers of the ESD cell pin shape. You can use these optional arguments to specify the top and bottom technology layers of the ESD cell pin shape. This provides you with the flexibility to specify the lowest layers to be used for creating pins in the LEF. In the following example, only the shapes between AP and M11 layers are ESD cell pin shapes:

Bbox file example:

```
ESD_diode_0 1281296 860923 1296972 895445 AP M11
ESD_diode_1 1281296 898651 1296972 933173 AP M11
```

Data_type: string, optional

`-extract_signal_net_logic_info_only`

Specifies to print only the logical connectivity for the signal nets in the `NETS` section of the output DEF file. If this parameter is not specified, the software prints both the logical connectivity and routing shapes for the signal nets in the `SPECIALNETS` section of the DEF file.

Data_type: bool, optional

`-stream_file` *filename*

Mandatory parameter. Specifies the name of the input GDS file. The top cell name in the GDSII file is the design name in the generated DEF.

Data_type: string, optional

`-stream_layer_map` *filename*

Mandatory parameter. The layer mapping file between GDS and tech layers. All the layers defined in a given mapping file is extracted. The layers not defined in the mapping file are ignored.

The format of the layer mapping file is:

`<layer_type> <tech layer name> <gds> <gds layer#> [gds layer data type] [PORT] [TEXT_ONLY] <mask keyword> [dummy]`

The format of the layer mapping file to identify the die area in the output DEF is:

`<layer> <prBoundary> <gds> <gds layer#> [gds layer data type]`

Format Description:

`<layer_type>` specifies the type of library layer. The various layer types are: `via`, `metal`, and `layer`.

`<tech layer name>` specifies the name of the library layer to map to the GDSII layer number. The library layers are technology layers which can be dumped using the `Techgen -techinfo` command.

`<gds>` specifies the keyword for the GDS layer mapping.

`<gds layer#>` specifies the layer number of the GDSII layer.

`[gds layer data type]` specifies the optional number of the subclass for the GDSII layer. If you do not specify a datatype, the software uses all datatypes for that GDSII layer. If GDS datatype is not defined, the software considers it as a regular layout routing layer where all the shapes will be extracted.

The GDS layer may have additional information/attribute assigned using the following keywords:

- `PORT`: Converts text labels in this layer to ports.
- `TEXT_ONLY`: Includes only the text in the named layer and ignores all geometries.

`<mask keyword>` keyword used to define the back-end metal layers with color masks (MASK A and Mask B).

`[dummy]` indicates that the layer is a metal fill layer. The `convert_stream_to_def` utility identifies these metal fill shapes and writes them to the `FILLS` section in the output DEF.

`<prBoundary>` specifies that the shapes on the `prBoundary` (Place-and-Route Boundaries) layer are used to determine the die area in the output DEF. This keyword can be used if you want to have the required die area in the output DEF file. If the `prBoundary` keyword is not specified, the die area is determined by the routing shapes specified in the GDS file, cell size, and the pin shapes of all the layers.

Example1:

```
metal METAL1 gds 15
via VIA1 gds 16
```

Example2:

The following is a sample layermap file with the mask keywords (`mask_a` and `mask_b`):

```
metal METAL1 gds 31 250
metal METAL1 gds 31 255 mask_a
metal METAL1 gds 31 256 mask_b
```

Here, the mask and main layers are defined using the same GDS layer number 31, but uses a different GDS data type (250, 255, and 256).

Example3:

The following is a sample layermap file with the `dummy` keyword:

```
metal  METAL5 gds 35 367 dummy
```

Here, the layer with the GDS layer ID `35` and data type `367` contains the metal fill shapes only.

Example4:

The following is a sample layermap file with the `prBoundary` keyword:

```
layer   prBoundary      gds     62 21
```

Data_type: string, optional

`-ignore_stream_inst_name`

Specifies to ignore the instance names from the specified GDS file, and the software will then automatically assign instance names to all the instances in the output DEF and Verilog files. This parameter can be used if instance names in the GDS file cannot be used, primarily in cases, such as duplicate instance names.

Data_type: bool, optional

`-inst_name_file filename`

Specifies the file name containing instance names for the generated DEF in the following format:

```
<x coord> <y coord> <instance name>
```

The x y coordinates are in the unit of micron. The priority of getting an instance name is:

1. instance name file
2. GDS label
3. internally assigned name

The `-inst_name_file` parameter allows to provide user-specified instance names. The `convert_stream_to_def` utility reads the GDS label to decide the instance name. If there are no labels, the software internally assigns instance names.

Data_type: string, optional

`-inst_name_prefix` *prefix*

Specifies a prefix for all the instance names in the output DEF and Verilog files.

Data_type: string, optional

`-net_location_file` *filename*

Mandatory parameter. Specifies the name of the net location file containing a list of pin locations for nets that needed to be considered. All the geometries of the P/G nets as well as the instance pins connected to the specified P/G nets are included in the `SPECIALNETS` section of DEF. The instances connected to the P/G nets are included in the `COMPONENTS` section of DEF.

The format of the net location file is:

```
<net name> <x loc of pin> <y loc of pin> <layer> <POWER|GROUND|SIGNAL> [pin name]
```

Example:

```
VDD 60 73 METAL3 POWER
VSS 75 40 METAL2 GROUND
```

Here,

- The x/y location in the file has the unit of micron.
- The layer name is the tech layer name.
- The type of a given net can only be POWER, GROUND, or SIGNAL. If the net type is marked as `SIGNAL`, the net will be traced from the GDS file and included in the `SPECIALNETS` section with "`USE SIGNAL`" in the output DEF file. In addition, the signal pin connectivity is specified for the net.
- The pin name field is optional. If there is no pin name field, the pin name will be same as the net name.

If you do not provide pin locations, the utility uses the GDS text labels in either the top-level cell or cells throughout the hierarchy to get the location. In such a case, the format of net location file is:

```
<net name> <POWER|GROUND|SIGNAL>
```

Example:

```
VDD POWER
VSS GROUND
```

Data_type: string, optional

`-output_components {ALL | NONE | CONN}`

Specifies the components to be printed in the DEF file `COMPONENTS` section during GDS to DEF conversion. The default is `ALL`.

- `ALL`: Specifies to output all the instances in the DEF Components section.
- `NONE`: Specifies not to print instances.
- `CONN`: Specifies to output instances with logical connectivity only.

Data_type: enum, optional

`-output_def_file` *filename*

Mandatory parameter. Specifies the name of the output DEF file. If the given DEF name is specified with "`.gz`", the software will gzip the DEF file to reduce the file size.

Data_type: string, optional

`-output_lef_macros` *filename*

Specifies the name of the file containing a list of macros for which LEF files are to be generated. One LEF file is generated for each macro. The generated LEF pin shapes are all shapes that connect to top-level nets on all metal/via layer shapes. The cell boundary of the LEF macro is derived by the bounding box of all shapes within the GDS cell.

Data_type: string, optional

`-power_grid_library` *libraryname*

Mandatory parameter. Specifies the name of the merged power-grid library (.cl). You can specify only a single .cl library.

Data_type: string, optional

`-print_complete_short_path`

Specifies to print the complete shorting path when there is a short between two nets. When this parameter is specified, the entire shorting path is assigned to one of the shorted nets and the path is printed in the generated DEF file. The shapes of the shorting path will be assigned to the net that is specified first in the net location file. For example, if a tracing point of VDD is shorted with a tracing point of VSS, the software honors the net that is defined first in the net location file.

If this parameter is not specified, the software assigns the shapes of the path to either of the shorted nets by default.

Data_type: bool, optional

`-spef_file` *filename*

Specifies the name of the SPEF file that can be used to include the signal nets in the generated DEF file. The input SPEF file is required to have layer annotation in the `RES` statements to indicate the pin node layers.

For the signal nets, both the `*P` and `*I` nodes are used as the starting tracing points, and the SPEF file pins `(*P)` are honored as the top-level pins and are reported in the `PINS` section in the output DEF file. For the PG nets, the `*P` nodes and the user-defined locations in the net location file are used as the starting tracing points. The SPEF file connections (`*I` statements) are honored when generating the DEF logic connectivity for both the signal and PG nets. That is, the `convert_stream_to_def` utility relies on the SPEF file for logic connection if a SPEF file is specified. The instance and pin names are from the SPEF `*I` statements. The output DEF file includes all the signal net pins from `*P` in the `PINS` section, the signal routing shapes in the `NETS` section, and the PG net routing shapes in the `SPECIALNETS` section.

In addition to the SPEF file, you need to specify the following parameters for including the signal nets in the DEF file:

- `-stream_file`
- `-net_location_file`
- `-power_grid_library`
- `-stream_layer_map`
- `-output_def_file`
- `-top_cell`

Notes:

- The `-net_location_file` parameter is used to identify the power and ground nets, and therefore the remaining nets are considered as signal nets.
- The `-power_grid_library` specifies the name of the power-grid library that includes the list of standard cells or macros that should be consistent with the SPEF file. That is, the cell of the instances in the SPEF file should be defined in the PGV.

Data_type: string, optional

`-top_cell` *cellname*

Mandatory parameter. Specifies the top cell name in the GDSII file.

`-use_stream_label {ALL | TOP | <`*tech_layer_name*`> | <`*cell_name*`>:<`*tech_layer_name*`> | <`*cell_name*`>:ALL>}`

	Specifies the criteria for selecting text labels. • `ALL`: specifies to honor all the text labels from GDS. • `TOP`: specifies to honor just the top-level text labels. • *<tech_layer_name>*: specifies to honor all the text labels on the specified layer. • *<cellname>*`:`*<tech_layer_name>*: specifies to honor all the text labels of the specified GDS cell on the specified layer. • *<cellname>*`:ALL`: specifies to honor all the text labels of the specified GDS cell. The following example specifies to use the text labels of the cell "hierblock" on layer METAL10: `convert_stream_to_def -use_stream_label hierblock:METAL10` *Data_type*: string, optional
`-verilog_file` *filename*	
	Specifies the name of the output verilog file. This parameter allows you to use the GUI to display the rail analysis results of the GDS-only flow. *Data_type*: string, optional
`-zero_pin_size`	
	Specifies to set the pin shape to size zero "(0 0)(0 0)" in the output DEF file. You can use this parameter when the port shape is outside the drawn shape (metal geometry) at places where the port label is defined near the edge of the geometry.

Examples

- The following command converts the specified GDS file `new.gds` to a DEF file `top.def`.
  ```
  convert_stream_to_def -net_location_file net.loc -stream_file new.gds -power_grid_library ../pv_library.cl -stream_layer_map lmap -top_cell cella -output_def_file top.def
  ```
- The following command converts the specified GDS file `chip.gds` to a DEF file `my.def`:
  ```
  convert_stream_to_def -stream_file chip.gds.gz -power_grid_library  {pgv/techonly/techonly.cl   pgv/full.cl} \
                    -net_location_file netlocfile -stream_layer_map  layermap -output_def_file my.def -top_cell chip -spef_file chip.spef.gz
  ```
  Here, the SPEF file `chip.spef` is specified to include the signal nets in the output DEF file.
- The following command specifies a marker layer file `esdcells` to enable ESD analysis using `convert_stream_to_def`:
  ```
  convert_stream_to_def -stream_file gw16_500_usr_test.gds  -power_grid_library { /../TECH/techonly.cl }  -net_location_file my.nlf -stream_layer_map  gds_layer.map \
      -output_def_file my.def  -top_cell gw16_500_usr -use_stream_label ALL -esd_marker_layer_file esdcells
  ```

  `convert_stream_to_def` utility generates a LEF file for each ESD cell specified in the `esdcells` file to enable PGV generation. The output DEF includes the user-defined nets and the ESD instances based on the specified marker layer file.

Related Information

- `set_rail_analysis_config`
- "GDS2DEF Utility" in the Voltus Stylus Common UI User Guide

generate_esd_rlrp_report

```
generate_esd_rlrp_report
[-help]
-net netname
[-instance_pair_list {{inst1 inst2 [pin1 pin2]}+} | -instance_pair_list_file filename]
[-number_threads value]
-state_directory directory
```

Reports the least-resistance path (RLRP) for the bump-to-clamp rule type in the ESD flow. This command helps to make debugging faster when a fail status is given for a bump-to-clamp pair. Using this command, you can:

- view the RLRP path for the selected bump-to-clamp pair in the GUI
- generate a point-to-point RLRP path report that displays the resistance between each node (resistor segment) pair along the RLRP path

Parameters

`-help`	Outputs the command usage and a brief description about the command parameters.
`-net netname`	
	Specifies the name of the net to be analyzed. *Data_type:* string, required
`-number_threads value`	
	Specifies the maximum number of threads to use when reporting the least-resistance path (RLRP) for the bump-to-clamp rule type in the ESD flow. Generally, using more threads decreases the run time. There is no absolute limit to the number of threads you can use, however, it is recommended that you do not specify more threads than the number of CPUs available. The default value is `1`, and the maximum value that can be specified is the total number of available CPUs. *Data_type:* int, optional
`-instance_pair_list {{inst1 inst2 [pin1 pin2]}+}`	
	Specifies the instance pair list to enable effective resistance calculation between two instances in the design. *pin* is an optional argument. By default, the tool will use the instance pin associated with the net being analyzed. The *pin* argument can be used in special cases when two different pins of an instance are connected to the same net. *pin1* and *pin2* are pins of *inst2* respectively. If a pin name is not given, all pins associated to the specified instance are used. *Data_type:* string, optional
`-instance_pair_list_file filename`	
	Specifies the name of the instance pair list file. This file contains the instance pair list in the same format as mentioned for the `-instance_pair_list` parameter. The file allows you to specify a large number of instance pairs. `-instance_pair_list_file` and `-instance_pair_list` are mutually exclusive. *Data_type:* string, optional
`-state_directory directory`	
	Specifies the name of the state directory in which the ESD analysis results are saved. *Data_type:* string, required

Examples

- The following commands specify to generate point-to-point RLRP path report for net `VDD_AO`:

```
generate_esd_rlrp_report \
-state_directory /rak_for_esd/ESD/ALL_125C_esd_1 \
-net VDD_AO \
-instance_pair_list { {VDD_AO_vsrc1 u0} {VDD_AO_vsrc2 u0 NA VDD} }
```

map_dies

```
map_dies
[-help]
-3dic_config_file filename
[-die_pad_file_pair {{<die_name1> <pad file1>} {<die_name1> <pad file2>}...}]
[-output_dir directory]
```

A utility to generate the inter die data mapping file based on the design stack configuration file and die ploc files. This mapping file is required if there is a direct connection between dies. The generated mapping file defines the connection between dies (between interposer and logic die). This mapping file is then specified with the `set_multi_die_analysis_mode -stacked_die_mapping_file` command for multi-die analysis.

Parameters

`-help`	Outputs the command usage and a brief description about the command parameters.
`-3dic_config_file filename`	
	Specifies the name of the XML file containing the design stack information. This file is from the Integrity™ 3D-IC product. It includes each die's tier level (Z direction), die instance name, size (post shrink size if there is a scale factor), flip, rotation (counter, clockwise), placement location, and contact layer.
`-die_pad_file_pair {{<die_name1> <pad file1>} {<die_name1> <pad file2>}...}`	
	Specifies the mapping between die instance names and their corresponding power pad file for the interposer die and logic dies.
`-output_dir directory`	

	Specifies the output directory containing the mapping results. The output folder contains the generated mapping file (`die_die.map`).
	The format of the 6-column mapping file is:
	`IDIE_INST_NAME(idie_inst_name) Inet_name Ibump_name`
	`LDIE_INST_NAME(ldie_inst_name) Lnet_name Lbump_name`
	Here,
	• The first 3 columns are for the interposer die instance name, net name, and bump name(vsrc name), and the last 3 columns are for the logic die instance name, net name, and bump name.
	• The die instance names are written in upper case that is followed by the die instance names specified in the rail TCL for each die instance name.
	• The file is split in two sections, POWER and GROUND.

Examples

- The following command generates a mapping file that includes the connection between dies (between interposer die INTERPOSER and logic dies DIE0, DIE1, DIE2, DIE3):

```
map_dies \
-output_dir results \
-3dic_config_file ../data/design_stack.xml \
-die_pad_file_pair { {INTERPOSER ../data/Interposer_all.ploc} \
                     {DIE0 ../data/Die0_all.ploc} \
                     {DIE1 ../data/Die1_all.ploc} \
                     {DIE2 ../data/Die2_all.ploc} \
                     {DIE3 ../data/Die3_all.ploc} }
```

Related Topics

- "Package Analysis" in the *Voltus User Guide*

report_differential_voltage

```
report_differential_voltage
[-help]
-differential_voltage_limit value
-power_directory directory_name
-state_directory directory_name
```

Determines the differential voltage between a receiver/driver pair during ramp-up analysis. This command can be used to mitigate the risk of high short-circuit current for a driver-receiver instance pair due to slow/fast (differential) ramp-up voltage at the output of the power-switch cell connected to the driver/receiver cell.

The `report_differential_voltage` command reports the differential potential for those receiver/driver pairs that are greater than the specified differential voltage threshold, and the time interval at which the differential voltage remains greater than the threshold. You must specify the `power_create_driver_db true` attribute before specifying this command.

The following is the format of the output report:

```
# <driverName>   <receiverName>   <peak_diffv>  <minT>   <maxT>
```

Parameters

`-help`	Outputs the command usage and a brief description about the command parameters.
`-differential_voltage_limit value`	
	Specifies to report differential voltage for the receiver/driver pairs that exceed the specified threshold limit.
`-power_directory directory_name`	
	Specifies the path to the power directory to access the driver-receiver pairs.
`-state_directory directory_name`	
	Specifies the path to the state directory generated by rail analysis to access the EIV or timing database.

Examples

- The following command lists all the driver-receiver pairs that exceed the differential voltage limit of 0.001:
  ```
  report_differential_voltage -differential_voltage_limit 0.001 -state_directory dynamic_rail/core_105C_dynamic_1 -power_directory dynamic_power
  ```
  The following is a snippet of the generated report:
  ```
  # <driverName>          <receiverName>       <peak_diffv>     <minT>     <maxT>
  column/MSV_I2         column/MULT8_reg_1_4    0.7382          5e-11    2.75e-09
  column/HIER_INST      column/MULT8_reg_1_4    0.7382          5e-11    2.75e-09
  ...
  ```

Related Topics

- "Static Power, IRdrop and EM Analysis" and "Dynamic Power and IRDrop Analysis" in the *Voltus User Guide*

report_irdrop_regions

```
report_irdrop_regions
[-help]
[-number_of_worst_instances value]
[-output_dir dir]
-state_directory dir
[-net netname | -domain domainname]
[-region {x1 y1 x2 y2 }| -number_of_region value]
[-eco_report]
[-net_threshold value | -domain_threshold value]
[-tile_row_column {X Y}]
[-tile_size {X Y}]
[-non_violating_insts_limit value]
[-enable_xp {true | false}]
```

Specifies to automatically identify the region with worst IRdrops and EM violations, and provide necessary information to determine and fix the root cause. The reports are consolidated for each region to find root-cause. Root cause analysis is done by identifying top 10 instances in each hotspot and report the following:

- Hotspot Region Coordinate
- Timing Library Analysis
- Power-Grid View Analysis
- Power-Grid Integrity Analysis
- IR-Drop Composition
- Root Cause Analysis

You must specify this command after running the `report_rail` command.

⚠ The `report_irdrop_regions` command requires a VTS-XL license.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `report_irdrop_regions` parameter. For a detailed description of the command and all of its parameters, use the man command man `report_irdrop_regions`.
`-domain domainname`	
	Specifies the name of the power domain. *Data_type*: string, required
`-domain_threshold value`	
	Specifies the minimum allowed voltage for the domain being analyzed. The value will be in percentage. This parameter behavior is same as the `set_rail_analysis_domain -threshold` parameter. *Data_type*: float, optional
`-eco_report`	
	Specifies to generate a file with IR drop information that can be used by Tempus ECO to fix IR voltage drop for all instances in a design. The report (`eco.ird`) is generated in the specified output directory (`report_irdrop_regions -output_dir`). *Data_type*: bool, optional
`-enable_xp {true | false}`	
	Specifies to identify the worst IR drop and EM violations regions in the Extensively Parallel (XP) mode wherein processing is distributed over a large number of machines. *Default* : `false` *Data_type*: enum, optional

`-net` *`netname`*	
	Specifies the name of the power or ground net that you want to analyze. *Data_type*: string, required
`-net_threshold` *`value`*	
	Specifies the minimum allowed voltage for the power net being analyzed. The value must be an absolute number. This parameter behavior is same as the `set_pg_nets -threshold` parameter. *Data_type*: float, optional
`-non_violating_insts_limit` *`value`*	
	Reports non-violating instances within hotspots (violating tiles) that are close to the threshold value. This parameter allows you to define a limit (sub-threshold) that can be used to report the non-violating instances that are close to the threshold. For example, if the `-net_threshold` or `-domain_threshold` is `60mV`, and `-non_violating_insts_limit` is `3mV` close to the threshold, the `report_irdrop_regions` command will report all the violating instances above the domain/net threshold and also the non-violating instances which are in the range of 60mV-57mV. This information about the non-violating instances can be used for IR improvement in the Timing-aware IR Drop Fixing flow. *Data_type*: float, optional
`-number_of_region` *`value`*	
	Specifies the number of regions to be analyzed. The default value is `3`. *Data_type*: int, optional
`-number_of_worst_instances` *`value`*	
	Specifies the number of worst instances to be reported. The default value is 5. *Data_type*: int, optional
`-output_dir` *`dir`*	
	Specifies the output directory in which the Irdrop and EM violation results will be saved in. *Data_type*: string, optional
`-region {`*`x1 y1 x2 y2`*` }`	
	Specifies the region containing the worst instances. The default unit of *x1 y1 x2 y2* is in database units. *Data_type*: rect, optional
`-state_directory` *`dir`*	
	Specifies the top-level state directory that is generated by the dynamic IR drop analysis. This state directory contains analysis of all nets associated with the power domain. This is the output directory in which the results of the `report_rail` command is stored (for example, core_25C_dynamic_1). *Data_type*: string, required
`-tile_row_column {X Y}`	
	Specify the number of tiles in the x and y direction. The `-tile_row_column` and `-tile_size` parameters enable tile-based analysis of the hotspot regions. Tile-based analysis divides the design into user-defined tile size, and reports IR drop violations on each tile. The default value is `100um^2 (10x10 tiles)`. *Data_type*: point, optional
`-tile_size {X Y}`	

	Specify the size of tiles in the x and y direction. The `-tile_row_column` and `-tile_size` parameters enable tile-based analysis of the hotspot regions. Tile-based analysis divides the design into user-defined tile size, and reports IR drop violations on each tile. *Data_type*: point, optional

```
Hot-Spot Debugger Report

* Top Worst Instances
ID       | Instance Name
--------------------------------
       1| inst_I2
       2| inst_I3
...

#Power-Grid Integrity Analysis
Instance | LEF Pin:PG Net            | Floating PG Pins
-------------------------------------------------------------
       1| VDD:VDD ; VSS:VSS         | None
       2| VDD:VDD ; VSS:VSS         | None
...

* Number of Weakly Connected Wire Segments
        VDD |         VSS
------------------------
          0 |           0

#Power-Grid View Library Status
Instance| Status | Power Pins(V)| Intrinsic Cap (fF)    | View Used  | #Current Taps | PGV Library Path
-------------------------------------------------------------------------------------------------
      1| PASS   | VDD:1.1      | VDD:13.480 VSS:13.480 | Early      | VDD:1 VSS:1 | /data/cl/stdcells.cl

...

#IR-Drop Composition
Instance| Total IR-drop (V)        | IR-drop on Power Net (V)| Ground-bounce on ground net (V)
-------------------------------------------------------------------------------------------
      1| 0.072 (TW) 0.072 (Worst) | 0.031 (TW) 0.031 (Worst)| 0.043 (TW) 0.043 (Worst)
...

* Layer Based IR-Drop for Net VDD
Instance| IR Across Package(mV) | IR Across Power-Gate(mV)| Layer Based IR(mV)
-------------------------------------------------------------------------------
       |                       |                         |         M5L       0.683
       |                       |                         |       C4L5L       0.477
 1     |  0.035                |  0.000                  |         M4L       0.394
-------------------------------------------------------------------------------
2 ...
-------------------------------------------------------------------------------

* Layer Based IR-Drop for Net VSS
...
-------------------------------------------------------------------------------

#Root Cause Analysis Report

* Net: VDD
Instance  | IR-Drop(V) | RLRP(Ohm) | Loading Cap(fF)| Current Density (mA/um2)| Transition Density| Frequency(Mhz) | Slew
-------------------------------------------------------------------------------------------------------------------
       1| 0.031*     | 66.5*     | 64.371*        | 0.022*                  | 615384576*        | 307*           | 0.100*
...
-------------------------------------------------------------------------------------------------------------------
Design Max| 0.058      | 159       | 140332.844     | 0.061                   | 615384640         | 307            | 0.984
Design Avg| 0.003      | 43.7      | 2.585          | 0.002                   | 93809752          | 300            | 0.076
Design Min| 0.000      | 0.0988    | 0.000          | 0.000                   | 0                 | 25             | 0.002

* Net: VSS
Instance  | IR-Drop(V) | RLRP(Ohm) | Loading Cap(fF)| Current Density (mA/um2)| Transition Density| Frequency(Mhz) | Slew
...
```

Examples

- The following command generates IR debug report for the `vss` net in the specified output directory:

  ```
  report_irdrop_regions -state_directory dynamic_rail/core_25C_dynamic_1/VSS -net VSS -output_dir IR_DEBUG_VSS -number_of_worst_instances
  10
  ```

Related Information

- `set_rail_analysis_config`
- "Static Power, IRdrop and EM Analysis" in the *Voltus Stylus Common UI User Guide*

report_noise_margin

```
report_noise_margin
[-help]
-noise_limit_file input_file
-power_directory dir
-state_directory dir
```

Specifies to calculate noise margins using the Effective Instance Voltage (EIV) method, that is, calculate noise margins with worst voltages seen on the driver-receiver pins. You need to specify the `power_create_driver_db true` attribute to generate the driver-receiver database.

To run this command, you need to specify the default or cell-specific noise limit, signal noise, guardband, rail directory, and power directory.

The command generates the noise margin output file in the state directory. The format of the EIV-based noise margin output file (*powernetname_groundnetname*`.noise_margin`) is:

Margin_High(V)	Margin_Low(V)	NMH (V)	NMI(V)	Voh(V)	Vol(V)	Vih(V)	Vil(V)	Signal_Noise(V)	Guardband(V)	Dr Inst/pin	Rx Inst/pin
NMH - signal noise - guardband	NMI - signal noise - guardband	Voh - Vih	Vil - Vol	VDD_driver - VSS_receiver	VSS_driver - VSS_receiver	(1-noise_limit) * (VDD_receiver - VSS_receiver) noise_limit is the noise limit value specified in the noise limit file	noise_limit* (VDD_receiver - VSS_receiver) noise_limit is the noise limit value specified in the noise limit file	Signal noise value specified in the noise limit file	Guardband value specified in the noise limit file	Driver pin name	Receiver pin name

Parameters

`-help`	Prints a brief description that includes the type and default information for each `report_noise_margin` parameter. For a detailed description of the command and all of its parameters, use the man command: `man report_noise_margin`
`-noise_limit_file` *input_file*	

	Specifies the name of the input noise limit file that contains the default or cell-specific noise limit, signal noise, and guardband. The format of the noise limit file is:
	`default_noise_limit <value>`
	`default_signal_noise <value>`
	`default_guard_band <value>`
	`<cell_name> <noise_limit> <signal_noise> <guard_band>`
	The following is a snippet of the noise limit file:
	`default_noise_limit 0.46`
	`default_signal_noise 0.1`
	`default_guard_band 0.05`

`-power_directory` *dir*	
	Specifies the path to the power directory to access driver-receiver pairs.
`-state_directory` *dir*	

	Specifies the path to the state directory generated by rail analysis to access the EIV or timing database.

Examples

- The following command calculates noise margin:

```
report_noise_margin -noise_limit_file ./noiselimit -state_directory dynamic_rail/core_105C_dynamic_1 -power_directory dynamic_power
```

Related Topics

- "Static Power, IRdrop and EM Analysis" in the *Voltus User Guide*

extract_package

```
extract_package
[-help] <domain>
[-output_dir <directoryname>]
-package_board_name <bga_circuitname>
-package_die_name <die_circuitname>
[-reference_net <netname>]
[-spd_file filename]
-workspace <workspace_name>
```

Specifies to start package model extraction.

Parameters

`-help`	Outputs the command usage and a brief description about the command parameters.
domain	Specifies the rail analysis domain name. *Data_type*: string, required
`-output_dir <directoryname>`	
	By default, the package model will be created in the workspace. However, if you specify an output directory, then a copy of the result will be made in the new directory. *Data_type*: string, optional
`-package_board_name <bga_circuitname>`	
	Defines the ball grid array (BGA) circuit name in the package model. It is the PCB component in the package model, which eventually connects to the PCB model. *Data_type*: string, optional
`-package_die_name <die_circuitname>`	
	Defines the die circuit name in the package model. When specified, the software automatically creates a mapping between the die name in the package extraction tool and the die name in the software. *Data_type*: string, optional
`-reference_net <netname>`	
	Specifies the ground net that is selected as reference net for pin-based model generation. This is the ground net that is being referred to by the power nets. *Data_type*: string, optional
`-spd_file filename`	
	Specifies an existing Sigrity .spd file for package extraction. This parameter allows you to specify a .spd file instead of the workspace name (package model database) of the package extraction tool (XtractIM). *Data_type*: string, optional
`-workspace <workspace_name>`	
	Specifies the workspace name (package model database) of the package extraction tool (XtractIM). The file name extension is `.ximx`. *Data_type*: string, optional

Example

- The following example specifies the package models details and starts extraction:

```
extract_package ALL \
  -workspace ../rak_xim_pkg_model/C1_Voltus_4L.ximx \
  -package_board_name FPBGA144 \
  -package_die_name CD1 \
  -output_dir rak_xim_pkg_model_voltus_cpf \
  -reference_net VSS
```

Related Information

- set_advanced_package_options
- "Package Analysis" in the *Voltus Stylus Common UI User Guide*

report_esd_network

```
 report_esd_network
[-help]
<run_name>
[-clamp_lowest_layer_taps {true | false}]
[-clamp_pin_short_file pin_shorting_file]
-config_file rule_filename
[-output_dir directory]
[-remove_layers_below layer_name]
[-ta | -task_assistant]
-type {net | domain}
[-use_power_pad {true | false}]
```

Performs different types of effective resistance checks, such as bump to clamp, bump to bump, and clamp to clamp, and current density analysis during an ESD zap event. This command checks whether every power or ground bump has an ESD device, and the device is placed in such a way that it does not violate the effective resistance or current density limit.

The objective of ESD Checker is to easily identify the violations caused by high effective resistance or the current density above the specified threshold value.

Parameters

`-help`	Outputs the command usage and a brief description about the command parameters.
`<run_name>`	Specifies the domain or net name. The result directory is prefixed with `<run_name>`. *Data_type*: string, required
`-clamp_lowest_layer_taps {true | false}`	
	Specifies to keep only the lowest layer of clamp taps for effective resistance analysis. The default value is `false`. • When set to `true`, the ESD pin nodes on the lowest layer are shorted, and effective resistance checks are performed to the shorted node on the lowest layer. • When set to `false`, the ESD pin nodes on all layers are shorted, and effective resistance checks are performed to the shorted pin nodes of clamps on different layers. *Data_type*: enum, optional
`-clamp_pin_short_file pin_shorting_file`	

	Specifies the list of pin layers to be shorted for bump-to-clamp and clamp-to-clamp resistance checks. If multiple layers for the same pin are specified, then these layers are shorted with each other.
	The format of the file is:
	`#Cell specification`
	`CELL <cell name> <pin> <layer>`
	`#Instance specification`
	`INST <instance name> <pin> <layer>`
	In this file, multiple layers for the same cell or instance can be specified by adding multiple rows for the same cell or instance name corresponding to each layer.
	If the `-clamp_pin_short_file` parameter is not specified, the software shorts all pin nodes of the lowest ESD cell pin layer, and performs effective resistance checks to the shorted node on the lowest layer.
	Note: If the `-clamp_pin_short_file` parameter is used in conjunction with `-clamp_lowest_layer_taps`, the `-clamp_pin_short_file` parameter will override the settings of `-clamp_lowest_layer_taps`.
	Data_type: string, optional

`-config_file rule_filename`	

	Mandatory parameter. Specifies the name of the rule file that includes all the ESD checks to be performed as a set of rules. It also specifies the effective resistance threshold limit. For more information, refer to the **ESD Rule Specification** section in the "ESD Analysis" chapter of *Voltus User Guide*. *Data_type*: string, required
`-output_dir` *directory*	
	Specifies the name of the output directory in which the ESD analysis results are saved. *Data_type*: string, optional
`-remove_layers_below` *layer_name*	
	Specifies the layer name below which all layers are dropped from analysis. This parameter can be used in scenarios where the ESD cells connect to the PG grid at high layers, and therefore the impact of the lower PG layers is very small on effective resistance calculation and current density checks. *Data_type*: string, optional
`-ta | -task_assistant`	
	Opens ESD Analysis Task Assistant, a task-based quick help functionality for the ESD analysis flow. The information in the Task Assistant is organized by topics relevant to the task you are performing. It answers frequently asked questions, such as format of the rule file, use model of the command, or supported ESD checks.
`-type {net | domain}`	
	Mandatory parameter. Specifies whether to run rail analysis on a domain or a net. *Data_type*: enum, required
`-use_power_pad {true | false}`	
	Specifies that the x,y locations of the bump cells are to be obtained from the power pad file. This parameter can be used to support designs for which bump cells are not visible. The default value is `false`. *Data_type*: enum, optional

Examples

- The following commands specify to run ESD analysis on the domain `top`:

```
set_rail_analysis_config -method static -accuracy hd -power_grid_libraries ./techonly.cl -ict_em_models ./Data/ESD_em.ict
report_esd_network top -config_file ./esd_new_rule.txt -type domain -output_dir ESD_analysis -use_power_pad true
```

Related Information

- `set_rail_analysis_config`
- "ESD Analysis" in the *Voltus Stylus Common UI User Guide*

report_esd_voltage

```
 report_esd_voltage
[-help]
-current_distribution_layer layer
-driver_receiver_file filename
-esd_pin_location_file filename
[-max_violations value]
-net NetName
[-report filename]
[-threshold value]
-total_current value
[-detailed_report {true | false}]
```

Performs IR Drop analysis for every driver/receiver pair and reports voltage drop above the given limit. Use this command to provide the current injection points (sinks), the current distribution layer (source), and the total ESD current to be distributed. The command generates a report containing all the violations where potential difference across driver/receiver is greater than the given threshold.

Parameters

`-help`	Outputs the command usage and a brief description about the command parameters.
`-current_distribution_layer layer`	
	Specifies to distribute current to all nodes of the given layer. This is the LEF/DEF layer name. This is a required parameter. *Data_type*: string, required
`-detailed_report {true | false}`	
	Specifies to report the differential voltage for each zap point above the specified threshold. It gives the x,y location of the zap point and the zap point name. Using this parameter, you can fix all the differential voltage violations in a single run. This parameter will generate the report file with the same name as specified using the `-report` parameter and with the `_detail.rpt` suffix. The following is an example of the parameter: `report_esd_voltage -net VSS -driver_receiver_file ./driverdb_railVSS -esd_pin_location_file VSS.pp -current_distribution_layer METAL5 -total_current 8 -report FULL_new_gnd_esd_cdm.rpt -detailed_report true -threshold 2` *Data_type*: string, optional
`-driver_receiver_file filename`	
	Specifies the name of the driver/receiver file of the net. This file is created by the `-create_driver_db` parameter of the `set_power_analysis_mode`. This is a required parameter. The file has multiple columns, wherein the first column is the receiver, and the remaining columns are drivers to the receiver instance. The following is an example of the file content: **`padring/I_RCV_7`** `io_toplevel/padring/I_DECAP io_toplevel/padring/` In this example: `padring/I_RCV_7` is the receiver instance. `io_toplevel/padring/I_DECAP` and `io_toplevel/padring/` are driver instances. *Data_type*: string, required
`-esd_pin_location_file filename`	

	Specifies the name of the ESD pin location file. This is a required parameter.
	The format of the file is:
	`#Pin_name x coord y coord layer`
	`gnd_pin_1 2284.6 -2898.05 m4`
	`...`
	`...`
	Data_type: string, required

`-max_violations value`	
	Specifies the number of max worst violations to be reported. This is an optional parameter. When this value is specified, only the specified number of top violations will be reported. If a value is not specified, voltage drop for all driver/receiver pairs will be reported. *Data_type*: int, optional
`-net NetName`	
	Specifies the name of the net to be analyzed. This is a required parameter. *Data_type*: string, required
`-report filename`	
	Specifies the name of the output report file in which details of the driver-receiver violation pair is written. This is an optional parameter. The report is filtered in descending order of differential voltage. The output file has four columns: • First column - `Differential voltage(V)` • Second column - `Driver name (node location(um) )` • Third column - `Receiver name (node location(um))` • Fourth column - `ESD pin name (location(nm))` The following is the sample ESD file format: `14.1324 pmc/a_ip_pmc_chassis_c55fg ( 2050.875 -2760.085 ap ) io_toplevel/padring/I_ESDCLAMP_39 ( 1428.200 2884.250 m4 ) gnd_pin_22 ( 1458.200 2898.050 m4 )` *Data_type*: string, optional
`-threshold value`	
	Specifies the minimum allowed driver/receiver voltage for filtering the report. This is an optional parameter. When a threshold value is specified, driver/receiver pairs with differential voltage greater than the specified threshold value will be reported. If a threshold value is not specified, voltage drop for all driver/receiver pairs will be reported. *Data_type*: float, optional
`-total_current value`	
	Specifies the total ESD current to be distributed. The default current unit is amperes. This is a required parameter. *Data_type*: string, required

Example

- The following example performs static IR Drop analysis to check the voltage drop between all driver and receiver pairs:
  ```
  report_esd_voltage -net gnd -driver_receiver_file ./POWER/driverdb_railgnd  -esd_pin_location_file ./epinfile -
  current_distribution_layer m1 -total_current 8 -report FULL_new_gnd_esd_cdm.rpt
  ```

Related Information

- `set_rail_analysis_config`
- `set_advanced_pg_library_mode`
- "ESD Analysis" in the *Voltus Stylus Common UI User Guide*

report_joule_heat

```
report_joule_heat
[-help]
[-detail_delta_temperature_file filename]
[-detail_delta_temperature_region {x1 y1 x2 y2}]
[-net pg_net_name]
[-power_domain rail_domain_name]
[-report_connected_pin_wire filename]
[-scale_rms_limit value]
[-tile_delta_temperature_file filename]
[-tiles {n_tile_in_x m_tile_in_y}]
[-output_directory dir_name]
```

Specifies to perform joule heat analysis. This command will use the dynamic RMS current to compute delta-T RMS on the power and signal wires for all segments in the design. The `report_joule_heat` command is similar to the `report_self_heat` command, except that the `report_self_heat` command computes both delta-T FEOL (FEOL self heat) and delta-T RMS (BEOL self heat) whereas the `report_joule_heat` command computes only delta-T RMS.

Joule heat analysis (`report_joule_heat`) should be performed for the entire design, that is, you must use domain analysis with all PG nets (`-power_domain ALL`).

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `report_joule_heat` parameter. For a detailed description of the command and all of its parameters, use the man command `man report_joule_heat`.
`-detail_delta_temperature_file filename`	
	Specifies the name of the generated BEOL detailed delta-temperature file. *Data_type*: string, optional
`-detail_delta_temperature_region {x1 y1 x2 y2}`	
	Specifies the region coordinates for generating detailed delta temperature file. *Data_type*: string, optional
`-net pg_net_name`	
	Specifies a net for joule heat analysis. *Data_type*: string, optional
`-power_domain rail_domain_name`	
	Specifies the domain name of the rail. *Data_type*: string, optional
`-output_directory dir_name`	
	Specifies the directory name for storing the joule heat analysis results. When you specify this parameter, the following result files are saved in the specified directory: • Rail analysis output directory • Detail delta temperature file If `-output_directory` is not specified, by default the joule heat analysis results are stored in the current working directory. *Data_type*: string, optional
`-report_connected_pin_wire filename`	

	Reports results on wires connecting to cell pins. *Data_type*: string, optional
`-scale_rms_limit` *value*	
	Specifies the scale factor for the root-mean-square (rms) limit relaxation factor during Electromigration current limit analysis. *Data_type*: float, optional
`-tile_delta_temperature_file` *filename*	
	Specifies the name of the generated BEOL tile-based delta-temperature file. *Data_type*: string, optional
`-tiles` *{n_tile_in_x m_tile_in_y}*	
	Specifies the number of tiles in the x and y directions in the tile-based delta temperature file. *Data_type*: string, optional

Examples

- The following command is an example of `report_joule_heat` command:
  ```
  report_joule_heat -power_domain ALL -tile_delta_temperature_file joule_ttm.txt -tiles {10 10} -detail_delta_temperature_file
  joule_heat_detail.rpt -report_connected_pin_wire joule_heat.rpt
  ```

Related Information

- `set_rail_analysis_config`
- `report_self_heat`
- "Self-Heating Effect Analysis" in the *Voltus Stylus Common UI User Guide*

report_package

```
report_package
[-help] <domain>
[-computer_resources string]
[-output_dir <directoryname>]
-package_board_name <bga_circuitname>
-package_die_name <die_circuitname>
-package_pin_current_file <filename>
-result_name <filename>
[-spd_file filename]
-type <package_analysis_type>
-workspace <workspace_name>
```

Specifies how the package analysis will be performed, and to run package analysis.

Parameters

`-help`	Outputs the command usage and a brief description about the command parameters.
domain	Specifies the rail analysis domain name. *Data_type*: string, required
`-computer_resources` *string*	
	Specifies the computer resources for the package analysis run. This parameter allows you to customize the host distribution/CPU usage for package analysis. The default setting is "`-smt 0 -local -cn localhost -cpus 8`". *Data_type*: string, optional
`-output_dir <`*directoryname*`>`	
	Specifies the output directory name of the package analysis results. By default, the package model will be created in the specified workspace. *Data_type*: string, optional
`-package_board_name <`*bga_circuitname*`>`	
	Defines the ball grid array (BGA) circuit name in the package model. *Data_type*: string, optional
`-package_die_name <`*die_circuitname*`>`	
	Defines the die circuit name in the package model. *Data_type*: string, optional
`-package_pin_current_file <`*filename*`>`	
	Specifies the name of the generated pin current file from rail analysis that contains the package pin names and the current values. It is automatically generated after the package model is created. *Data_type*: string, optional
`-result_name <`*filename*`>`	
	Specifies the name of the .xml result file. *Data_type*: string, optional
`-spd_file` *filename*	
	Specifies an existing Sigrity .spd file for package analysis. This parameter allows you to specify a .spd file instead of the workspace name (package model database) of the package analysis (PowerDC) tool. *Data_type*: string, optional
`-type <`*package_analysis_type*`>`	
	Specifies the type of package analysis. Currently, the software supports only DC package analysis. *Data_type*: enum, required
`-workspace <`*workspace_name*`>`	
	Specifies the workspace name for package analysis (powerDC). The file name extension is `.pdcx`. *Data_type*: string, optional

Example

- The following example specifies the package analysis details and starts package analysis:

```
report_package ALL \
-type dc \
-workspace /../sigrity/rak_pdc_irdrop/test.pdcx \
-package_board_name FPBGA144_Signal02 \
-package_die_name CD1_Signal01 \
-output_dir rak_pdc_irdrop_voltus_cpf \
-result_name rak_pdc_irdrop_voltus \
-package_pin_current_file /../spa_cpf/ALL_125C_avg_1/Reports/super_filter_pkg_iv.rpt
```

Related Information

- set_rail_analysis_config
- view_package_results
- "Package Analysis" in the *Voltus Stylus Common UI User Guide*

report_rail

```
report_rail
[-type {domain | net}]
[-output_dir directory_name]
name
```

Runs rail analysis on a net or domain. This command must be specified after defining the rail analysis mode properties using the `set_rail_analysis_config` command.

Parameters

name	Specifies the name of the domain or net that you want to analyze. *Data_type*: string, required
`-output_dir directory_name`	
	Specifies the name of the directory in which to store the output directories created during the analysis. *Default*: Current working directory. *Data_type*: string, optional
`-type {domain | net}`	
	Specifies whether to run rail analysis on a domain or a net. A domain is a list of power or ground nets to analyze together. See `set_power_pads` for more details. You can specify the variable "ALL" to analyze all domains. *Data_type*: enum, required

Examples

- The following command performs an analysis for the vdd net and places the results directories in the analysis_out directory:
  ```
  report_rail -type net -output_dir analysis_out vdd
  ```
- The next command performs rail analysis on TDSP power domain and writes the results in current working directory.
  ```
  report_rail -type domain -output_dir ./ TDSP
  ```
- The following command performs rail analysis on all power domains and writes the results in the current working directory:
  ```
  report_rail -type domain ALL -output_dir ./
  ```

Related Information

- `set_rail_analysis_config`
- "Static Power, IRdrop and EM Analysis" in the *Voltus Stylus Common UI User Guide*
- "Dynamic Power and IRDrop Analysis" in the *Voltus Stylus Common UI User Guide*

report_resistance

```
report_resistance
[-help]
[-cell_name master_cell_name]
[-die_mode {3dic}]
[-force_reextraction {true | false}]
[-instance_list {instance1 pin1 instance2 pin2...}]
[-instance_list_file filename]
[-instance_pair_list {instance1 instance2 pin1  pin2 ...}]
[-instance_pair_list_file filename]
-net_name netName | -domain_name domainName
[-node_list {{nodeX nodeY METAL_LAYER [LABEL]}...}]
[-node_list_file filename]
[-node_pair_list {node1X node1Y METAL_LAYER1 node2X node2Y METAL_LAYER2...LABEL}]
[-node_pair_list_file filename]
[-output_file filename]
[-output_dir directory_name]
[-region {x1 y1 x2 y2 layer number_of_nodes region_name}]
[-report_limit value]
[-threshold value]
[-write_cell_name]
[-domain_threshold value]
[-net_threshold value]
[-exclude_list_file filename]
```

Performs effective resistance analysis. Effective resistance computes effective resistance between two nodes on the power-grid, or from a node to all the voltage sources in the design, or from an instance to all the voltage sources in the design.

You can run this command without performing power calculation and IRdrop analysis. The flow requires you to load the design, set power nets, set power pads, and run resistance analysis. This command enables the tool to consider the entire power structure, including embedded ring structure within IO cells if any.

Note: The `report_resistance` full chip analysis now supports running the `set_rail_analysis_mode -enable_rlrp_analysis` command to perform least resistance path analysis (RLRP) in conjunction with effective resistance analysis (REFF).

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `report_resistance` parameter. For a detailed description of the command and all of its parameters, use the man command `man report_resistance`.
`-cell_name master_cell_name`	
	Reports effective resistance for all instances of the specified master cell name. The `-cell_name` parameter improves the usability of effective resistar you to specify only the cell name of the instances you want to analyze. You can specify multiple cells for performing effective resistance analysis of *Data_type*: string, optional
`-die_mode {3dic}`	
	Specifies to perform multi-die effective resistance analysis.
`-domain_name domainName`	
	Specifies the domain name that is to be analyzed. *Data_type*: string, optional
`-domain_threshold value`	
	Specifies the minimum allowed effective resistance (effective resistance sum of each net) to instance. For specific instance list analysis, the instanc will be reported as “PASS”, while the instances above the threshold value will be reported as “FAIL”. For full chip analysis, the instances below the reported in the domain report. The default value is 0. *Data_type*: float, optional
`-exclude_list_file filename`	

Specifies the name of a file containing a list of cells or instances that will not be processed during the resistance analysis flow. The specified cells/ir report generation.
The use model of this parameter is:
`report_resistance -exclude_list_file <file_name>`

File Format:
`<CELL | INST> <name>`

Example:
```
CELL BUMP_MASTER
CELL ESD_MASTER
INST INSTA
```

Data_type: string, optional

`-force_reextraction {true | false}`

Specifies to force a re-extract of resistances.

Data_type: enum, optional

`-instance_list {instance1 pin1 instance2 pin2...}`

Specifies the list of instances for which effective resistance is to be calculated. *pin* is an optional argument. By default, the tool will use the instance being analyzed. The *pin* argument can be used in special cases when two different pins of an instance are connected to the same net.

Note: The `-instance_list` and `-instance_list_file` parameters are mutually exclusive to the `-instance_pair_list` and `-instance_pair_list_fil` that you can specify `-instance_list` and `-instance_list_file` together but you cannot specify `-instance_list` and `-instance_list_pair` together.

Data_type: list, optional

`-instance_list_file filename`

Specifies the name of the instance list file. This file contains the list of instances in the same format as mentioned for the `-instance_list` parameter. specify a large number of instances. `-instance_list_file` and `-instance_list` are mutually exclusive.

Note: The `-instance_list` and `-instance_list_file` parameters are mutually exclusive to the `-instance_pair_list` and `-instance_pair_list_fil` that you can specify `-instance_list` and `-instance_list_file` together but you cannot specify `-instance_list` and `-instance_list_pair` together.

Data_type: string, optional

`-instance_pair_list {instance1 instance2 pin1 pin2 ... }`

Specifies the instance pair list to enable effective resistance calculation between two instances in the design. `pin1` and `pin2` are pins of `instance1`

Note: The `-instance_list` and `-instance_list_file` parameters are mutually exclusive to the `-instance_pair_list` and `-instance_pair_list_fil` that you can specify `-instance_list` and `-instance_list_file` together but you cannot specify `-instance_list` and `-instance_list_pair` together.

Data_type: list, optional

`-instance_pair_list_file filename`

Specifies the name of the instance pair list file. This file contains the instance pair list in the same format as mentioned for the `-instance_pair_list` you to specify a large number of instance pairs. `-instance_pair_list_file` and `-instance_pair_list` are mutually exclusive.

Note: The `-instance_list` and `-instance_list_file` parameters are mutually exclusive to the `-instance_pair_list` and `-instance_pair_list_fil` that you can specify `-instance_list` and `-instance_list_file` together but you cannot specify `-instance_list` and `-instance_list_pair` together.

Data_type: string, optional

`-net_name netName`

Specifies the net name that is to be analyzed.

Data_type: string, optional

`-net_threshold value`

The effective resistance requirements for different power nets vary depending on the design. For example, an array or periphery supply to an SRAM lower effective resistance threshold compared to the core supplies. This parameter allows you to specify different threshold values for different nets.

The use model of this parameter is given below:

```
set_rail_analysis_domain -domain_name ALL -power_nets {vdd vddar vdda vdd1 vdd2} -ground_nets vss
report_resistance -threshold 50 -net_threshold {{vdd 50} {vddar 20} {vdda 35} {vss 40}} -domain_name ALL -output_dir Reffective
```

In this example, the threshold value for different nets are:

vdd -> 50
vddar → 20
vdda -> 35
vss -> 40
vdd1 -> 50
vdd2 -> 50

Here, for all the remaining nets for which a value is not specified using the `-net_threshold` parameter, the value specified with the `-threshold` parar

The different threshold values will have an impact on the list of instances reported in the `<state_dir>/Reports/<net>/<net>.effr` report.

Data_type: list, optional

`-node_list {{nodeX nodeY METAL_LAYER [LABEL]}...}`

Specifies the list of user-defined nodes for effective resistance calculation between the specified user-defined nodes and all voltage sources. You ca on the region on a given layer. `LABEL` is any name chosen by you to identify node entries in the list. This is optional but helps during debugging.

Note: The `-node_list` and `-node_list_file` parameters are mutually exclusive to the `-node_pair_list` and `-node_pair_list_file` parameters. This `-node_list` and `-node_list_file` together but you cannot specify `-node_list` and `-node_pair_list` together.

Data_type: list, optional

`-node_list_file filename`

Specifies the name of the node list file. This file contains the list of user-defined nodes in the same format as mentioned for the `-node_list` paramete specify a large number of nodes.

You can specify multiple nets for node list based resistance analysis. The use model for multiple nets is:
`report_resistance -domain <domain name> -node_list_file <filename>`

Here,

- the `-domain` parameter will specify the domain name, that is, the domain with nets that is required to be analyzed.
- the `-node_list_file` parameter will include the multiple net names

The format for specifying multiple nets is:

```
NET <netname1>
node1X node1Y METAL_LAYER [LABEL]
node2X node2Y METAL_LAYER [LABEL]

NET <netname2>
node1X node1Y METAL_LAYER [LABEL]
node2X node2Y METAL_LAYER [LABEL]
```

Note: The `-node_list` and `-node_list_file` parameters are mutually exclusive to the `-node_pair_list` and `-node_pair_list_file` parameters. This `-node_list` and `-node_list_file` together but you cannot specify `-node_list` and `-node_pair_list` together.

Data_type: string, optional

`-node_pair_list {node1X node1Y METAL_LAYER1 node2X node2Y METAL_LAYER2...LABEL}`

Specifies the node pair list for effective resistance calculation between two nodes on the power-grid. `You can add a list of nodes based on the LABEL` is any name chosen by you to identify node entries in the list. This is optional but helps during debugging.

Note: The `-node_list` and `-node_list_file` parameters are mutually exclusive to the `-node_pair_list` and `-node_pair_list_file` parameters. This `-node_list` and `-node_list_file` together but you cannot specify `-node_list` and `-node_pair_list` together.

Data_type: list, optional

`-node_pair_list_file filename`

Specifies the name of the node pair list file. This file contains the node pair list in the same format as mentioned for the `-node_pair_list` parameter. a large number of node pairs.

You can specify multiple nets for node pair list based resistance analysis. The use model for multiple nets is:
`report_resistance -domain <domain name> -node_pair_list_file <filename>`

Here,

- the `-domain` parameter will specify the domain name, that is, the domain with nets that is required to be analyzed.
- the `-node_pair_list_file` parameter will include the multiple net names

The format for specifying multiple nets is:

```
NET <netname1>
node1X node1Y METAL_LAYER1 node2X node2Y METAL_LAYER2 [LABEL]
node3X node3Y METAL_LAYER3 node4X node4Y METAL_LAYER4 [LABEL]

NET <netname2>
node1X node1Y METAL_LAYER1 node2X node2Y METAL_LAYER2 [LABEL]
node3X node3Y METAL_LAYER1 node4X node4Y METAL_LAYER4 [LABEL]
```

Note: The `-node_list` and `-node_list_file` parameters are mutually exclusive to the `-node_pair_list` and `-node_pair_list_file` parameters. This `-node_list` and `-node_list_file` together but you cannot specify `-node_list` and `-node_pair_list` together.

Data_type: string, optional

`-output_file filename`

Specifies the name of the output report file that contains the report header and the sorted list for resistance values.

Data_type: string, optional

`-output_dir directory_name`

Specifies the name of the directory in which the output report file is saved.

Data_type: string, optional

`-region {x1 y1 x2 y2 layer number_of_nodes region_name}`

Analyzes resistance in the specified region. You can draw a box on a specific layer, and specify the number of nodes to be selected (maximum limit parameter allows you to compute effective resistance on a given layer across the chip and determine how the resistance is changing for the design can assign a name for the region.

Data_type: string, optional

`-report_limit value`

Specifies the number of instances printed in the report to save disk space.

Data_type: int, optional

`-threshold value`

Specifies the minimum allowed effective resistance to display node/node-pair/instance/instance-pair that have a resistance value above this thresh threshold as a non zero value, then instances with resistance value above that threshold will be tagged as FAIL and the rest will be tagged as PASS

Data_type: float, optional

`-write_cell_name`

Specifies to print cell names in the effective resistance report. This parameter allows you to sort resistance values by cell names.

Data_type: bool, optional

Examples

- The following command performs effective resistance calculation for the list of nodes and instances:
  ```
  report_resistance -net_name VSS -node_list { 231 1061.6 M1 } -instance_list_file instance.list
  ```
- The following command specifying a list of user-defined nodes and a node list file for effective resistance calculation:
  ```
  report_resistance -net_name VDD -node_list {{ 213.556 133.469 M1  I123 }} -node_list_file /../node_list.txt.1 -output_file reff_rpt
  ```
- The following is an example of specifying a list of instances for effective resistance calculation:
  ```
  report_resistance -instance_list {{INV11 VDD} {INV21 VDD}} -net_name VDD
  ```
- The following is an example of specifying a list of user-defined nodes for effective resistance calculation:
  ```
  report_resistance -net_name VDD -node_list {{800 11.88 M4} {999.768 11.88 M4} {1000.296 11.880 M4}} -output_file out.txt
  ```
- The following is an example of specifying node pair list for effective resistance calculation:
  ```
  report_resistance -net_name VDD -node_pair_list {195.728 133.369 M1 213.556 133.469 M1} -output_file out_nodelist.txt
  ```
- The following is an example of specifying instance pair list for effective resistance calculation:
  ```
  report_resistance -net_name GND -instance_pair_list {inst1 inst2 GND GND} -output_file outfile -threshold 100
  ```
- The following command performs effective resistance calculation for full chip analysis:
  ```
  report_resistance -threshold 13.0  -domain_threshold 25.0 -domain_name ALL
  ```

  Sample Report:

  ```
  NET: VSS VDD
  # Threshold:   25.000
  # Total Effective resistance for instances sorted from high to low
  # PASS/FAIL         REFF  INSTANCE                 PIN...

    FAIL            25.664  clk__L1_I1       VDD(51.52%)      VSS(48.48%)
    FAIL             25.35  clk__L2_I1       VDD(52.16%)      VSS(47.84%)
    FAIL            25.037  clk__L3_I1       VDD(51.56%)      VSS(48.44%)
  ```

- The following command performs effective resistance calculation for instance list analysis:
  ```
  report_resistance -domain_name ALL -threshold 13.0 -domain_threshold 25.0 -instance_list_file inst.list
  ```

  Sample Report:

  ```
  NET: VSS VDD
  # Threshold:   25.000
  # Total Effective resistance for instances sorted from high to low
  # PASS/FAIL         REFF  INSTANCE             PIN...
    FAIL            25.664  clk__L1_I1       VDD(51.52%)      VSS(48.48%)
    FAIL            25.35   clk__L2_I1       VDD(52.16%)      VSS(47.84%)
    FAIL            25.037  clk__L3_I1       VDD(51.56%)      VSS(48.44%)
    PASS            24.724  clk__L4_I1       VDD(52.21%)      VSS(47.79%)
    PASS            23.047  inst_pll         VDD(50.55%)      VSS(49.45%)
  ```

- The following command reports reff for all instances of the cell type CKBD8:
  ```
  report_resistance -net_name VDD -cell_name CKBD8
  ```
  The following is a sample reff report generated using the `-cell_name` parameter:

  ****

  ```
  # Layer Representation: LEF
  # Coordinate Unit: um
  # Resistance Unit: Ohm
  NET: VDD
  # Threshold:    0.000
  # Effective resistance for instances sorted from low to high
  # PASS/FAIL       REFF        INSTANCE          PIN
    -              3.5752    add_94_37_I2/g695    VDD
    -              3.7521    add_94_37_I2/g694    VDD
  ```

Related Information

- `set_rail_analysis_config`
- "Static Power, IRdrop and EM Analysis" in the *Voltus Stylus Common UI User Guide*

report_self_heat

```
report_self_heat
[-help]
[-power_domain rail_domain_name]
[-output_directory   dir_name]
```

Runs Self-Heating Effect (SHE) analysis on a single die or multiple dies. This command must be specified after defining the SHE properties using the `set_self_heat_analysis_mode` command.

The `report_self_heat` command will use the static power data to compute delta T FEOL for all instances in the design. This command:

- computes the RMS currents on signal wires
- computes the RMS currents on power network wires
- generates Detail Delta Temperature File (ddt), Instance Delta Temperature File (idt), and Tile Delta Temperature File (tdt)

ⓘ Self-heat analysis (`report_self_heat`) should be performed for the entire design, that is, you must use domain analysis with all PG nets (`-power_domain ALL`).

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `report_self_heat` parameter. For a detailed description of the command and all of its parameters, use the man command `man report_self_heat`.
`-power_domain rail_domain_name`	
	Specifies the domain name of the rail.
`-output_directory   dir_name`	
	Specifies the directory name for storing the self-heat analysis results. When you specify this parameter, the following result files are saved in the specified directory: • Rail analysis output directory • Detail delta temperature file • Instance delta temperature file • Tile delta temperature file If `-output_directory` is not specified, by default the self-heat analysis results are stored in the current working directory.

Examples

- The following command is an example of `report_self_heat` command in the multi-die analysis flow:

```
set_self_heat_analysis_mode \
-die_instance_name die0 \
-alpha_parameters {{M1 0.74 0.85} {M2 0.62 0.85} {M3 0.53 0.85} ....} \
-beta_parameters {0.00423 -0.00045 0.81534}\
-instance_power_file ./dynamic_power.rpt \
-cell_thermal_resistance_file TRF.txt \
-instance_delta_temperature_file sh_itm.txt \
-tile_delta_temperature_file sh_ttm.txt \
-tiles {10 10} \
-detail_delta_temperature_file power_detail_die0.rpt

set_self_heat_analysis_mode \
-die_instance_name die1 \
-alpha_parameters {{M1 0.74 0.85} {M2 0.62 0.85} {M3 0.53 0.85} ...} \
-beta_parameters {0.00423 -0.00045 0.81534}\
-instance_power_file ./dynamic_power.rpt \
-cell_thermal_resistance_file TRF.txt \
-instance_delta_temperature_file sh_itm.txt \
-tile_delta_temperature_file sh_ttm.txt \
-tiles {10 10} \
-detail_delta_temperature_file power_detail_die1.rpt

set_self_heat_analysis_mode \
-die_instance_name die2 \
-alpha_parameters {{M1 0.74 0.85} {M2 0.62 0.85} {M3 0.53 0.85} ...} \
-beta_parameters {0.00423 -0.00045 0.81534}\
-instance_power_file ./dynamic_power.rpt \
-cell_thermal_resistance_file TRF.txt \
-instance_delta_temperature_file sh_itm.txt \
-tile_delta_temperature_file sh_ttm.txt \
-tiles {10 10} \
-detail_delta_temperature_file power_detail_die2.rpt

report_self_heat -power_domain core
```

Related Topics

- "Self-Heating Effect Analysis" in the *Voltus User Guide*

report_signal_resistance

```
report_signal_resistance
[-help]
[-instance_pair_list_file filename]
-net net_name
[-output_dir directory]
[-report filename]
[-threshold value]
```

Computes effective resistance for signal nets from a given instance1/pin1 to instance2/pin2 accounting all applicable manufacturing effects. You can use the effective resistance output file to determine whether the critical routes are within the specified resistance limits.

Parameters

`-help`	Outputs the command usage and a brief description about the command parameters.
`-instance_pair_list_file filename`	
	Specifies the file name containing the signal net instance-pin pairs for which effective resistance needs to be computed. The format of the file is: `INST1 PIN1 INST2 PIN2` *Data_type*: string, optional
`-net net_name`	
	Specifies the name of the signal net. *Data_type*: string, required
`-output_dir directory`	
	Specifies the name of the output directory. *Data_type*: string, optional
`-report filename`	
	Specifies the name of the report file in which details of effective resistance between instance-pin pairs are written. *Data_type*: string, optional
`-threshold value`	
	Specifies the minimum allowed effective resistance between instance-pin pairs. *Data_type*: float, optional

Examples

- The following command computes effective resistance for the signal net `n_13190`:
  ```
  report_signal_resistance -net n_13190
  ```
- The following command computes effective resistance for the signal net `n_13190`, and writes the effective resistance details in the report `n_13190.rpt`:
  ```
  report_signal_resistance -net n_13190 -report n_13190.rpt
  ```
- The following command computes effective resistance for signal net instance-pin pairs specified in the file `instance.list`:
  ```
  report_signal_resistance -net n_0 -instance_pair_list_file instance.list
  ```

Related Information

- `set_rail_analysis_config`
- `report_resistance`
- "Static Power, IRdrop and EM Analysis" in the *Voltus Stylus Common UI User Guide*

scale_what_if_capacitance

```
scale_what_if_capacitance
[-help]
-reset | {{-global | -region x1 y1 x2 y2 | -cell cell_name | -instance inst_name | -file filename}
[-intrinsic_capacitance value]
[-loading_capacitance value]
[-grid_capacitance value]
[-add]
[-layer name]
[-net net_name]
}
```

Scales the intrinsic cell capacitance value to determine how much additional capacitance can be added in a region to reduce dynamic IR drop.

Parameters

`-add`	
	Specifies to add capacitance (farads) on the power-grid on a specific layer and region. The parameter works in conjunction with the existing parameters, such as `-intrinsic_capacitance`, `-grid_capacitance`, and `-loading_capacitance`. When this parameter is specified, the value provided with the `-intrinsic_capacitance`, `-grid_capacitance`, and `-loading_capacitance` parameters will be treated as absolute value instead of a scale factor value, which is the default behavior. *Data_type*: bool, optional
`-cell cell_name`	
	Specifies to scale the capacitance value for the specified cell. *Data_type*: string, optional
`-file filename`	

	Specifies a file containing the list of instances and cells that use the same scaling factor. The format of the file is:
	`CELL cellname`
	`INSTANCE instancename`
	Data_type: string

`-global | -region x1 y1 x2 y2`	
	You must specify either `-global` or `-region x1 y1 x2 y2`. The `-global` option scales the specified capacitances for the entire design. The `-region x1 y1 x2 y2` scales the specified capacitances in the region. The default unit of `x1 y1 x2 y2` is in database units. *Data_type for -global*: bool, required *Data_type for -region*: string, required
`-instance inst_name`	
	Specifies to scale the capacitance value for the specified instance. *Data_type*: string, optional
`-layer layername | all`	

	Specifies the LEF layer on which capacitance scaling is performed. Multiple commands can be issued to scale capacitance on multiple layers. You can specify the `all` argument to scale all the layers. *Data_type*: string, optional
`-net` *net_name*	
	Specifies to scale the capacitance value for the specified net. *Data_type*: string, optional
`-help`	Outputs a brief description that includes the type and default information for each `scale_what_if_capacitance` parameter. For a detailed description of the command and all of its parameters, use the man command `man scale_what_if_capacitance.`
`-reset`	Resets all the previous `scale_what_if_capacitance` command settings. *Data_type*: bool, optional
`-grid_capacitance` *value*	
	Scales power-grid capacitance of the design to the specified scale factor. The default scale value is 1. *Data_type*: string, optional ⚠ The `-grid_capacitance` parameter can be specified either with the `-global` or the `-region` parameter. This parameter is not supported with the `-cell` and `-instance` parameters.
`-intrinsic_capacitance` *value*	
	Scales instance internal gate capacitances to the specified scale factor. The default scale value is 1. *Data_type*: string, optional
`-loading_capacitance` *value*	
	Scales loading capacitance of the instance to the specified scale factor. The default scale value is 1. *Data_type*: string, optional

Examples

- **Cell-based capacitance scaling** - The following command scales intrinsic capacitance on all instances of the cell *cellA* by 1e-10 (<*intrinsic_cap*> * 1e-10):
```
scale_what_if_capacitance -cell cellA  -intrinsic_capacitance 1e-10
```
- **Pin-based capacitance scaling** - The following command scales intrinsic and loading capacitance on all instances of the cell *cellA* connected to the pin *vdd* by 1e-10 (<*intrinsic_cap*> * 1e-10 and <*loading_cap*> *1e-10):
```
scale_what_if_capacitance -cell cellA -net vdd -intrinsic_capacitance 1e-10  -add -loading_capacitance 1e-10
```
- **Region-based capacitance scaling** - The following command scales capacitance in the specified region by 1e-10 (<*intrinsic_cap*> * 1e-10, <*loading_cap*> *1e-10, and <*grid_cap*> * 1e-10):
```
scale_what_if_capacitance -region {896.877  937.661  1276.9365  1615.358} -intrinsic_capacitance 1e-10  -loading_capacitance 1e-10  -grid_capacitance 1e-10
```
- The following command scales grid capacitance in the specified layer by 1e-10:
```
scale_what_if_capacitance -net VDD -region { 0 0 100 100 } -layer M2 -grid_capacitance 1e-10
```

Related Information

- `scale_what_if_resistance`
- `scale_what_if_current`
- `write_current_region`
- "What-If Rail Analysis" in the *Voltus Stylus Common UI User Guide*

scale_what_if_current

```
scale_what_if_current
[-help]
[-hierarchy hier_name | -region x1 y1 x2 y2 | -global | -instance_list filename]
[-scale_clock_network]
[-scale value | -current value]
[-net net_name]
[-reset]
```

Scales current to do what-if analysis for a hierarchical partition and assess its effect on IR drop.

Parameters

`-current value`	
	Specifies current value in Amperes. This is used as target peak current for the given hierarchy and is then scaled appropriately. You cannot specify this option with -region *x1 y1 x2 y2* or -global options. *Data_type*: string, optional
`-global`	Specifies current scaling to be applied to entire design. *Data_type*: bool, optional
`-help`	Outputs a brief description that includes the type and default information for each `scale_what_if_current` parameter. For a detailed description of the command and all of its parameters, use the man command man `scale_what_if_current`.
`-hierarchy hier_name`	
	Specifies the name of the hierarchical block. For example, -hierarchy top/blockA. To specify global scaling for the design use "/". For example, -hierarchy / `-hierarchy` is a required parameter. This parameter allows instance-based current scaling by the specified scale factor. The parameter can be used to scale down peak current values. The use model of instance current scaling is: `scale_what_if_current -hierarchy <instance_name> -scale <value>` *Data_type*: string, optional
`-net net_name`	
	Scales current of the specified net. *Data_type*: string, optional
`-instance_list filename`	

	Specifies to scale current for the specified nets. This parameter allows you to specify a file containing the list of instances and their scale factor.
	Use Model
	`scale_what_if_current -instance_list filename -scale val`
	The format of `filename` is:
	`instanceName scaleFactor`
	`instanceName scaleFactor`
	Here,
	• `val` means the global scale factor. This value will be used if the local or instance-specific scale factor is not specified.
	• `scaleFactor` means the local scale factor. If both global and local scale factors are specified, the local scale factor takes precedence.
	Default scale factor value is `1`.

`-region x1 y1 x2 y2`	

	Specifies a region where current scaling is to be applied. The default unit of x and y coordinates is in microns (um). *Data_type*: string, optional
`-reset`	Resets all the previous `scale_what_if_current` command settings. *Data_type*: bool, optional
`-scale` *value*	
	Specifies any value greater than 0. For example, a value of 0.5 will scale the current in half. *Data_type*: string, optional
`-scale_clock_network`	
	When specified, scales current for all instances within the given hierarchy. When not specified, the software does not scale the instances that belong to the clock network. It scales current for the rest of the instances within the given hierarchy to the specified value. You cannot specify this option with -region *x1 y1 x2 y2* or -global options. *Data_type*: bool, optional

Examples

- The following command scales current for the hierarchical block `BLOCK_C`:
```
scale_what_if_current -hierarchy top/BLOCK_C -current 0.5
```

Related Information

- `scale_what_if_capacitance`
- `scale_what_if_resistance`
- `write_current_region`
- "What-If Rail Analysis" in the *Voltus Stylus Common UI User Guide*

scale_what_if_resistance

```
scale_what_if_resistance
[-help]
-reset | {-net netname
{-global | -region x1 y1 x2 y2 | -instance instancename | -cell cellname | -file filename}
[-scale value | -scale_ron {R_ON_scale_factor | R_OFF}]
[-auto_scale_adjacent_via_layers {true | false}]
[-layer {layername | all} ]
}
```

Scales resistance globally or in a region to determine how much resistance can be modified in a region to reduce the entire IR drop. In designs with power switches, the scaling of resistance will be applied to always-on net as well as its switched nets in a given region. Currently, it is not possible to apply different scaling factors for always-on and switched nets.

Parameters

`-auto_scale_adjacent_via_layers {true|false}`	
	Specifies to use auto via scaling for what-if resistance analysis. When scaling resistance on a specific metal layer, via resistance on the adjacent via layers are automatically scaled with the same scaling factor. The default value is true. Use `-auto_scale_adjacent_via_layers false` to keep via resistances un-scaled. *Data_type*: enum, optional
`-global | -region x1 y1 x2 y2 | -instance instancename | -cell cellname | -file filename`	

	You must specify either `-global`, `-region`, `-cell`, `-instance`, or `-file`.
	• When the `-global` parameter is specified with the `-scale` parameter, it scales resistance by the specified value globally for a given layer. If the `-global` parameter is specified with the `-scale_ron` parameter, it scales the resistance value of all power switch cells globally for a given layer. This is the default.
	• When the `-region` parameter is specified with the `-scale` parameter, it scales the grid resistance in the specified region. If the `region` parameter is specified with the `-scale_ron` parameter, it scales the resistance of all power switch cells in the specified region. The default unit of x1 y1 x2 y2 is in database units.
	• The `-instance instancename` or `-cell cellname` parameter specifies to scale the internal resistors from the PGV for the specified instance or cell.
	• The `-file filename` parameter specifies a file containing the list of instances and cells that use the same scaling factor. The format of the file is:
	`CELL cellname`
	`INSTANCE instancename`
	This is a required parameter. Default is `-global`.
	The precedence for resistance scaling is as follows:
	1. Instance
	2. Cell
	3. Region
	4. Global
	Data_type for -global: bool, required
	Data_type for -region: string, required
	Data_type for -instance: string, required
	Data_type for -cell: string, required
	Data_type for -file: string, required

`-help`	Outputs a brief description that includes the type and default information for each `scale_what_if_resistance` parameter. For a detailed description of the command and all of its parameters, use the man command man `scale_what_if_resistance`.

`-layer` *layername* | `all`	
	Specifies the layer name (LEF layer or technology layer) on which the scaling is performed. Multiple commands can be issued to scale resistance on multiple layers. You can specify the `all` argument to scale all the layers. When the `all` argument is used, the `-auto_scale_adjacent_via_layer` parameter is disabled. *Data_type*: string, required
`-net` *netname*	
	Scales resistance of the specified net in the domain which is being analyzed. *Data_type*: string, required ⚠ Specify always-on net name when scaling powergate on resistance.
`-reset`	Resets all the previous `scale_what_if_resistance` command settings. *Data_type*: bool, optional
`-scale` *value*	
	Specifies any value greater than 0. For example, a value of 0.5 will scale the resistance in half. *Data_type*: string, optional
`-scale_ron {`*R_ON_scale_factor* | `R_OFF}`	
	Specifies the value used to scale the RON of the power-gate instance/cell. The default value is `1`. You can also specify the "`R_OFF`"(case insensitive) keyword to apply the off state power-gate resistance for the specified instance/cell. *Data_type*: string, required

Related Information

- `scale_what_if_capacitance`
- `scale_what_if_current`
- `write_current_region`
- "What-If Rail Analysis" in the *Voltus Stylus Common UI User Guide*

set_advanced_package_options

```
set_advanced_package_options
[-help]
-reset  |
{[-tool_path <binary_path>] [-net_mapping_file <filename>]}
```

Specifies the path to the bin folder of package extraction and analysis tools. Also, a net mapping file needs to be specified. The mapping file is a simple two column format containing the die net name and the corresponding package net names. After running this command, use the `extract_package` command to start package model extraction.

Parameters

`-help`	Outputs the command usage and a brief description about the command parameters.
`-tool_path <binary_path>`	
	Specifies the full path to the bin folder of the package extraction and analysis tools (XtractIM and PowerDC). *Data_type*: string, optional
`-net_mapping_file <filename>`	
	Specifies the name of the two column mapping file that contains the mapping between the net names of package with the net names of die. *Data_type*: string, optional
`-reset`	
	Resets all specified options back to default values. *Data_type*: bool, optional

Example

- The following example specifies the path to the bin folder of package extraction and analysis tools:

```
set_advanced_package_options \
   -tool_path /home/sigrity/tools/bin/ \
   -net_mapping_file net_mapping.txt
```

Related Information

- `extract_package`
- "Package Analysis" in the *Voltus Stylus Common UI User Guide*

set_advanced_rail_options

```
set_advanced_rail_options
[-help][-reset] | [[-voltus_rail_include_file_begin file_name1]
[-voltus_rail_include_file_end file_name2] ]
```

Provides the ability to include a file that provides a list of additional rail options that should be used for rail analysis. If you do not specify values to the set_advanced_rail_options command parameters, the parameter values are set to the default value, that is, empty file name.

For more information on the `vstorm2` commands, refer to the "Rail Analysis Options" section in the *Voltus IC Power Integrity Solution Text Command Reference.*

Parameters

`-help`	Outputs the command usage and a brief description about the command parameters.
`-reset`	Resets parameters to their default values. This means that an extra include file is not required. *Data_type*: bool, optional
`-voltus_rail_include_file_begin file_name`	
	Provides the ability to provide a file that includes additional rail options that will be run prior to the `analyze` command. This can be used for those legacy rail analysis commands that do not have a Voltus equivalent. *Data_type*: string, optional
`-voltus_rail_include_file_end file_name`	
	Provides the ability to provide a file that includes additional rail options that will be run after the `analyze` command. This can be used for those legacy rail analysis commands that do not have a Voltus equivalent. *Data_type*: string, optional

Examples

- The following command needs to be set to override the default precision digits:
  ```
  set_advanced_rail_options -voltus_rail_include_file_begin vstorm.file


  <vstorm.file>
  setvar precision_digits 10
  ```
  With this variable, you will get IR with 10 precision digits (10 includes all digits before and after the decimal point).

Related Information

- `set_rail_analysis_config`
- "Static Power, IRdrop and EM Analysis" in the *Voltus Stylus Common UI User Guide*
- "Dynamic Power and IRDrop Analysis" in the *Voltus Stylus Common UI User Guide*

set_die_model

```
set_die_model
[-help]
[-sub_circuit_name subcircuitname]
{-reset  | [[-die_instance_name dieinstname [-spice_model_file model_file [-mapping_file mapping_file]]] ]
}
```

Specifies to import a die model on die Interposer or on package netlist.

This command gives information about the die model that will be used in the rail analysis of another die, wherein the two die are connected through die Interposer die or package netlist. For example, if you are performing IR drop analysis for a die (**die2**) on an Interposer/package design that is connected to another die (**die1**), and if the two die share the power/ground network, it is imperative to import the die model of **die1** during IR drop analysis of **die2**.

You can import multiple die models by specifying multiple `set_die_model` commands.

Parameters

`-die_instance_name` *dieinstname*	
	Mandatory parameter. Specifies the die instance name for die model. This is the die model of the die that needs to be imported. *Data_type*: string, optional
`-help`	Outputs a brief description that includes the type and default information for each `set_die_model` parameter. For a detailed description of the command and all of its parameters, use the man command `man set_die_model`.
`-mapping_file` *mapping_file*	

	Mandatory parameter. Specifies the name of the mapping file that contains the port connection mapping between the die model and voltage source of the chip to be analyzed.
	The two mapping file formats are:
	• **Mapping when die model on LEF/DEF-based Interposer Die**
	`#port_of_die_model cell die_name/VSRC_pad_name`
	`VDD_Bump_VDD_9_0 cell Interposer/Bump_DIE0_VDD_0`
	`VDD_Bump_VDD_9_0 cell Interposer/Bump_DIE0_VDD_1`
	...
	• **Mapping when die model on Package Design**
	`#port_of_die_model pkg port_of_pkg_netlist`
	`VDD_Bump_VDD_9_0 pkg VDD_3`
	`VDD_Bump_VDD_9_0 pkg VDD_4`
	Data_type: string, optional

`-reset`	Resets all the previous `set_die_model` command settings to the default settings. *Data_type*: bool, optional
`-spice_model_file` *model_file*	
	Mandatory parameter. Specifies a file containing a SPICE subcircuit describing the die model. *Data_type*: string, optional
`-sub_circuit_name` *subcircuitname*	
	Defines the top-level subcircuit in package files. When this parameter is not specified, the first circuit in the Spice file is used as the default top circuit. *Data_type*: string, optional

Examples

- The following command imports two die models `Die1` and `Die2`:

```
set_die_model -die_instance_name Die1 -spice_model_file die1_model.ckt -mapping_file die1_port.txt
set_die_model -die_instance_name Die2 -spice_model_file die2_model.ckt -mapping_file die2_port.txt
```

Related Information

- `set_package`
- `write_die_model`
- "Package Analysis" in the *Voltus Stylus Common UI User Guide*

set_dynamic_rail_simulation

```
set_dynamic_rail_simulation
[-start_time value]
[-stop_time value]
[-resolution value]
[-reset]
[-auto_repeat]
```

An optional command to set the parameters for dynamic rail analysis. By default, dynamic rail analysis uses the start time, stop time, and resolution from current data file. To analyze fewer clock cycles, you can choose different start and stop times using the `-start_time` and `-stop_time` options. It is recommended that at least one or two full cycles are simulated to fully charge and discharge the on-chip capacitance.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `set_dynamic_rail_simulation` parameter. For a detailed description of the command and all of its parameters, use the man command man `set_dynamic_rail_simulation`.
`-auto_repeat`	

	Specifies to repeat the dynamic current waveform till the `-stop` time specified. If `-stop` is not specified, all the waveforms will repeat till the longest waveform duration. The `-auto_repeat` parameter can be used when multiple dynamic current files have different simulation periods or when IR drop analysis needs to be run for longer duration.
	The following command sets the dynamic simulation time from 20ns to 100ns in which the current data will be repeated until 100ns is reached:
	`set_dynamic_rail_simulation -start 20ns -auto_repeat -stop 100ns`
	The `-repeat` parameter of the `set_power_data` command has higher priority than `set_dynamic_rail_simulation -auto_repeat`, as shown in the following example:
	`set_power_data VDD1.ptiavg -repeat 50`
	`set_power_data VDD2.ptiavg`
	`set_dynamic_rail_simulation -auto_repeat -stop 100`
	In this example, VDD1 waveform will repeat till 50ns, while VDD2 waveform will repeat till 100ns.
	Data_type: bool, optional

`-reset`	Resets all the previous `set_dynamic_rail_simulation` command settings to the default settings. *Data_type*: bool, optional
`-resolution value`	
	By default rail analysis picks up the resolution (or step size) saved in the dynamic current files. To achieve better rail analysis performance, you can increase the step size using this option. However, using larger than two times the step size of current data file is not recommended as it can impact accuracy of dynamic IR drop analysis. You can specify the units with the value. If you do not specify a unit, the software uses the default unit. *Default*: Picoseconds (ps). *Data_type*: float, optional
`-start_time value`	
	Defines the start time to be used for dynamic rail analysis. This option is used with the `-stop_time` option and enables you to perform a more detailed analysis in specific time ranges. You can specify the units with the value. If you do not specify a unit, the software uses the default unit. *Default*: The default unit is nanoseconds (ns). The default value is the start time from current data file. The start time in the current data file is usually 0ns. *Data_type*: float, optional
`-stop_time value`	

	Defines the stop time to be used for dynamic rail analysis. This option is used with the `-start_time` option and enables you to perform a more detailed analysis in specific time ranges. You can specify the units with the value. If you do not specify a unit, the software uses the default unit. *Default*: The default unit is nanoseconds (ns). The default value is the end of the current data. If multiple current files are specified and they all have different simulation periods, the largest simulation period is used to decide the stop time. *Data_type*: float, optional

Examples

- The following command sets the dynamic simulation time from 2ns to 12ns with a resolution of 100ps:

```
set_dynamic_rail_simulation -start_time 2ns -stop_time 12ns -resolution 100ps
```

Related Information

- `set_rail_analysis_config`
- `set_power_data`
- "Dynamic Power and IRDrop Analysis" in the *Voltus Stylus Common UI User Guide*

set_multi_die_analysis_mode

```
set_multi_die_analysis_mode
-analysis_name analysis_name
-die_list {die_instance_name1 die_instance_name2}
[-report_rlrp_path_instance_list filename]
[-stacked_die_mapping_file Innovus_map_file]
```

Specifies how the multi-die analysis will be performed.

Parameters

`-die_list` *die_instance_name1 die_instance_name2*
Specifies the die instance names. *Data_type*: string, required
`-analysis_name` *analysis_name*
Specifies the name of the analysis. *Data_type*: string, required
`-report_rlrp_path_instance_list` *filename*

Specifies a file containing a list of die instances (one per line) for which the RLRP path should be reported. This parameter is used to generate the RLRP path across multiple die.

The format of the specified input file is:
`<die_name> <net_name>`

The `-report_rlrp_path_instance_list` parameter generates a report `mdie_inst_rlrp_path.rpt` at the following location:
`<output_dir>/<state_dir>/results/rail/.`

The format of the report is:

```
Die Name: <die_name>
Net Name: <net_name>
Pin Name: <pin_name>
Instance Name: <inst_name>
Die   Layer/Via   Resistance(cumulative)   Ohm Length   Node1#(X,Y)
Node2#(X,Y)   Net
```

Example of an Output File:

```
Die Name: LDIE1
Net Name: VDD_AO
Pin Name: TVDD
Instance Name: ring/PSO_RING_78

Die Layer/Via    Resistance(cumulative) Ohm Length Node1#(X,Y) Node2#
(X,Y) Net
LDIE1   METAL_1   0.1238(3.4182)           2.210   169.28500,233.27500
167.07500,233.27500 VDD_AO
LDIE1   METAL_1   0.1302(2.7137)           2.035   169.28500,221.89000
169.28500,223.92500 VDD_AO
LDIE1   VIA_1     0.0269(2.5835)           0.140   169.28500,221.89000
169.28500,221.89000 VDD_AO
LDIE1   METAL_2   0.0070(2.5565)           0.145   169.28500,221.74500
169.28500,221.89000 VDD_AO
LDIE1   VIA_2     0.0875(2.5496)           0.140   169.28500,221.74500
169.28500,221.74500 VDD_AO
LDIE1   VIA_3     0.0875(2.4621)           0.140   169.28500,221.74500
169.28500,221.74500 VDD_AO
...
```

Data_type: string, optional

`-stacked_die_mapping_file` *Innovus_map_file*

Specifies the name of the inter die data mapping file from Innovus. The mapping file is a virtual mapping that provides the connections between any 2 stacked die with communication directly between the die (master/slave configuration).

Data_type: string, optional

Examples

- The following command specifies an analysis name called `tsv`, two die instances called `mother_die` and `daughter_die`,and a stacked mapping file `pwr.mapping`:

```
set_multi_die_analysis_mode -analysis_name tsv -die_list {mother_die daughter_die}
-stacked_die_mapping_file ../data/pwr.mapping
```

Related Information

- `set_rail_analysis_config`
- `write_die_model`
- "Through-Silicon Via and System-in-Package" in the *Voltus Stylus Common UI User Guide*

set_net_group

```
set_net_group
-reset | {-name group_name
-type [power | ground]
-nets {net_list}
}
```

Specifies the list of either power or ground nets that are isolated but need to be analyzed together as a single net. The results are generated for the new grouped net. The new grouped net name must be one of the net names in the group. To analyze the grouped net, you must also specify power pads for the grouped net name.

Parameters

`-name group_name`	
	Name of the net group. **Note:** The name of the net group must be one of the net names in the group. Using a net name that is not in the net group will trigger error when displaying results in the flow. *Data_type*: string, required
`-nets {net_list}`	
	Specifies a list of power or ground nets. *Data_type*: string, required
`-reset`	Resets all previously created net name(s) with the new grouped net name(s). *Data_type*: bool, optional
`-type [power | ground]`	
	Specifies whether it is a power or ground net. *Data_type*: string, required

Examples

- The following example groups the power nets that will be analyzed together as a group called VDD.

```
set_net_group -name VDD -type power -nets { VDD VDD1 VDD2 VDD3 }
set_power_pads -net VDD -format xy -file VDD.pp
report_rail -type net VDD
```

Related Information

- `set_rail_analysis_config`
- `set_power_pads`
- "Static Power, IRdrop and EM Analysis" in the *Voltus Stylus Common UI User Guide*
- "Dynamic Power and IRDrop Analysis" in the *Voltus Stylus Common UI User Guide*

set_off_chip_package_trace

```
set_off_chip_package_trace
[-help]
-mapping_file mapping_file
-name switched_name
-pad_location_files list_of_bump_location_files
[-reset]
-spice_model_file spice_model_name
[-sub_circuit_name subcircuitname]
```

Specifies to include RDL0 (off-chip package trace) resistance effects during the static and dynamic rail analysis. This flow allows you to analyze switched net bumps that are marked as RDL0 bumps. When this command is specified, the software automatically generates switched net bump current and bump voltage waveforms in the `voltus_rail_pad.tran.ptiavg` file created in the new switch net directory located under the rail state directory.

You can specify multiple `set_off_chip_package_trace` commands to define connectivity of different bump nets.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `set_off_chip_package_trace` parameter. For a detailed description of the command and all of its parameters, use the man command `man set_off_chip_package_trace`.
`-mapping_file mapping_file`	
	Specifies the name of the mapping file that contains the mapping between the package port and switched net bump names. The format of the file is: `<spice_circuit_port_name> pin <vsrcName_in_switch_net's_ploc>` *Data_type*: string, required
`-name trace_model_name`	
	Specifies the trace model name or switched net bump of RDL0. *Data_type*: string, required
`-pad_location_files list_of_bump_location_files`	

	Specifies the name of a file containing switched net names and associated ploc/power pad files. The format of the file is:
	`SWITCHED_NET1 plocfile1`
	`SWITCHED_NET2 plocfile2`
	…..
	The format of plocfile (power pad) is:
	`<voltage_source_name> <x> <y> <metal_layer>`
	The format of this file is same as specified for the `set_power_pads -format xy` format.
	Data_type: string, required

`-reset`	Resets all the previous `set_off_chip_package_trace` command settings to the default settings. *Data_type*: bool, optional
`-spice_model_file model_file`	
	Specifies a file containing a SPICE subcircuit (RDL0 netlist or full-chip package netlist) for off-chip package trace. *Data_type*: string, required
`-subckt subcircuitname`	
	Defines the top-level subcircuit for off-chip package tracing. *Data_type*: string, optional

Examples

- The following command specifies to analyze the switched net `VDDau`:

```
set_off_chip_package_trace -name VDDau  -pad_location_files  pad_location_files.txt -spice_model_file pkg.ckt -mapping_file
pkg_mapping.file
```

Related Information

- `set_package`
- `write_die_model`
- "Package Analysis" in the *Voltus Stylus Common UI User Guide*

set_package

```
set_package
[-help]
-reset | {-spice_model_file model_file
[-mapping_file mapping_file]
[-offset {x y}]
[-die_instance_name dieinstname]
[-rotate_angle rotation_angle_in_degrees]
[-flip {true | false}]
[-sub_circuit_name subcircuitname]
[-scale scaling_factor_number]
[-mapping_type { distributed | lumped }]
}
```

Specifies the package model information to be used in power rail analysis.

Parameters

`-die_instance_name dieinstname`	
	Specifies the die instance name for multi-die analysis. This option is only available in multi-die analysis (`set_rail_analysis_mode -die_mode multi-die`). *Data_type*: string, optional
`-flip {true | false}`	
	Specifies to flip the die along the Y axis. Default value is `false`. *Data_type*: enum, optional
`-help`	Outputs a brief description that includes the type and default information for each `set_package` parameter. For a detailed description of the command and all of its parameters, use the man command `man set_package`.
`-mapping_file mapping_file`	
	Specifies the name of the package terminal mapping file, which contains the mapping of the terminals of the SPICE subcircuit in the package model file (Specified by the `-spice_model_file model_file` parameter) to the locations at the grid or top-level pins. The mapping file formats supported by the software are: • Model Connection Protocol (MCP) • Voltus-specific Format For more information on the mapping file formats, refer to the *Package Model and Mapping File* section in the "File Formats" chapter of *Voltus User Guide*. *Data_type*: string, optional
`-mapping_type { distributed | lumped }`	

	Specifies the type of mapping to be used in the Voltus-specific terminal mapping file that will be added in the rail analysis flow. The possible arguments are:
	• `distributed` – A distributed mapping file lists out the mapping for each bump on the die to its corresponding package port. The mapping file is specified with the `set_package -mapping` parameter.
	• `lumped` - A lumped mapping file allows for auto-grouping of the die bumps to the lumped package ports. When `lumped` is specified, the software implements a lumped package by internally creating a lumped package model and its mapping file.
	
	The format of the lumped mapping file is:
	`<terminal_name>  <die | vsrc>  <net_name>`
	`vddcore_in  die  vddcore`
	`vddcore_out  vsrc  vddmpu`
	
	In this format:
	◦ the software connects "`<terminal_name> die <net_name>`" to the bumps belonging to `<net_name>` on the die
	◦ the software connects "`<terminal_name> vsrc <net_name>`" to the voltage source of `<net_name>`
	◦ The number of lines in the lumped mapping file are few and are equal to the number of terminals on the package model
	Default: `distributed`
	Data_type: string, optional

`-offset {x y}`	
	Specifies to adjust and shift the X Y coordinates in the MCP header for each pin. When performing package analysis, the MCP is used to map the terminals of the SPICE subcircuit in the package model file to the locations at the grid or top-level pins in DEF. The die origin specified in MCP must be same as specified in DEF. If the die origin is not the same, you can use the `-offset` parameter to modify the XY location specified in the MCP file. *Data_type*: point, optional
`-reset`	Resets all the previous `set_package` command settings to the default settings. *Data_type*: bool, optional
`-rotate_angle` *rotation_angle_in_degrees*	
	Specifies the rotation angle of the die. The possible values are: `0/45/90/135/180/225/270/315`. Rotation is always in the counter clockwise direction from the original point. Default value is `0`. This information is required to do coordinate conversion for xy based pin mapping. *Data_type*: float, optional
`-scale` *scaling_factor_number*	
	Specifies to scale the package model. *Data_type*: float, optional
`-spice_model_file` *model_file*	
	Specifies a file containing a SPICE subcircuit describing the package model. *Data_type*: string, optional
`-sub_circuit_name` *subcircuitname*	

	Defines the top level subcircuit in package files. *Data_type*: string, optional

Related Information

- `write_die_model`
- "Package Analysis" in the *Voltus Stylus Common UI User Guide*
- "File Formats" in the *Voltus Stylus Common UI User Guide*

set_pg_nets

```
set_pg_nets
-net net_name
-voltage value
[-threshold value]
-tolerance value
-force
[-die_instance_name dieinstname]
[-lib_binding_voltage value]
[-package_net_name package_net_name]
[-package_voltage_drop_limit value]
```

Specifies the power grid nets. Using this command, you can specify power/ground net voltages in the static and dynamic power analysis flow, and the rail analysis flow.

For the power analysis flow, you must specify the `set_pg_nets` command before `report_power`.

Note: When specifying a ground net with a non-zero nominal operating voltage value, a threshold greater than the ground value must be specified in order for the net to be treated as the ground net.

Parameters

`-die_instance_name dieinstname`	
	Specifies the die instance name for multi-die analysis. This option is only available in multi-die analysis (`set_rail_analysis_mode -die_mode mult` *Data_type*: string, optional
`-force`	Skips the validity check on power nets. *Data_type*: bool, optional
`-help`	Outputs a brief description that includes the type and default information for each `set_pg_nets` parameter. For a detailed description of the command and all of its parameters, use the man command `man set_pg_nets`.
`-lib_binding_voltage value`	
	Specifies the voltage to be used for library binding in power analysis. This parameter binds the instances to the library that is closest to the user-sp voltage.
`-net net_name`	
	Specifies the name of the power or ground net that you want to analyze. *Data_type*: string, required
`-package_net_name package_net_name`	
	Specifies the net name on the package side. By default, the package net name is same as the net name. *Data_type*: string, optional
`-threshold value`	
	Specifies the minimum allowed voltage on the power net or the maximum rise on the ground net, depending on the setting of the `-voltage` parame parameter is used to verify that the power grid passes the limit check and to set the filters for an IR plot. *Data_type*: float, optional
`-tolerance value`	

	Specifies the tolerance range of the voltage value. Specify a value from 0 to 1 (not including 1). A value of 0.2 results in a range of "voltage +20%" *Default*: 0.3. *Data_type*: float, optional
`-voltage value`	
	Specifies the voltage on the power rail to be analyzed. *Data_type*: float, required
`-package_voltage_drop_limit value`	
	Specifies the voltage drop limit for a net. This parameter allows you to specify the voltage threshold value for each net separately. The specified tl the rail reports, histograms, and plots. If the threshold value is not specified for some of the nets being analyzed, the default range will be applied tc *Default Range*: 0mV to max VU *Data_type*: float, optional

Examples

- The following command specifies a VDD_TDSPCore value of 0.9 volts with a threshold of 0.89V and a tolerance of 0.3:
  ```
  set_pg_nets -net VDD_TDSPCore -voltage 0.9 -threshold 0.890 -tolerance 0.3
  ```
- The following command specifies to use the library binding voltage for binding the libraries to the instances instead of "`-voltage`":
  ```
  set_pg_nets -net vdd -voltage 0.90 -lib_binding_voltage 0.75
  ```

Related Information

- `set_rail_analysis_config`
- `set_rail_analysis_domain`
- "Static Power, IRdrop and EM Analysis" in the *Voltus Stylus Common UI User Guide*
- "Dynamic Power and IRDrop Analysis" in the *Voltus Stylus Common UI User Guide*

set_power_data

```
set_power_data
-reset | {-format {current | ascii | area | ascii_current}
[-instance_name instance_name]
[-offset offset_value]
[-power value]
[-scale factor]
[-repeat_time time]
[-bias_voltage value]
[-die_instance_name dieinstname]
[filename]
[-power_output_directory dir]
[-repeat_window {start end}]
[-strip_hierarchy_instance_name instance]
}
```

Specifies the static and dynamic power or current files for IRdrop and EM analysis. The power data for the design can be generated using the `report_power` command. This command also has ability to read external ASCII static power files to do static rail analysis. For dynamic analysis, you must use the `report_power` command to generate dynamic current files.

This command should be specified prior to the `report_rail` command. You must specify the `set_power_data` command after the `report_power` command in the flow script if power data is not available.

Parameters

`-bias_voltage value`	
	Specifies the value of the bias voltage. This parameter must be set if -format is set to ascii or area. *Data_type*: float, optional
`-die_instance_name dieinstname`	
	Specifies the die instance name for multi-die analysis. This option is only available in multi-die analysis (`set_rail_analysis_mode -die_mode m` *Data_type*: string, optional
filename	Specifies the name of the power data files. The file is optional if -format area is specified. *Data_type*: string, optional
`-format {current | ascii | area | ascii_current}`	
	Specifies the format of the power data. The power data format can be `current, ascii`, or `area`. The parameter `-bias_voltage` must be set if `area` or `ascii` is specified. • When using the current format option, specify the current file (*.ptiavg) that is generated by the power analysis software within Voltus. For exa `set_power_data -format current static_VDD.ptiavg`

	• When using the `ascii` format option, specify the ascii instance based power file, generated by the Power Calculation engine within Voltus. F
	`set_power_data -format ascii -bias_voltage 1.08 powermeter.pwr`
	Note: The command accepts an ASCII power file in three-column format consisting of *instance/cell name*, *power* in W,and *power pin name*. are treated as comment lines.
	The format of the three-column input file is as follows:
	`<instance/cell name> <power in Watts> <power pin name>`
	Example:
	`top/inst1 0.001 vdd`
	`top/inst2 0.001 vdd1`
	`top/inst3 0.003 vdd3`
	where vdd, vdd1, and vdd3 are power pin names of the instance
	ⓘ Although `-bias_voltage` is a required argument with `-format ascii`, when 3-column power file format is specified, the program will ignore rely on associated net voltages to determine current for the pins.

	• When using the `area` format option, specify the total power of the chip in Watts. This power is distributed based on cell area. Optionally, you for macro cells using a file, for example, `set_power_data -format area -bias_voltage 1.08 -power 10 cell_power.file` where, `-bias_voltage` is the voltage at which power was characterized; If the analysis voltage is different, this power will be scaled accordingly. `cell_power.file` is in two column format with *`cell_name`**power_in_watts*.
	• When using the `ascii_current` format option, specify an ASCII instance current file for static analysis. The format of the three-column input fi *`<instance/cell name> <current in Ampere> <power pin name>`* *Data_type*: enum, required
`-help`	Outputs a brief description that includes the type and default information for each `set_power_data` parameter. For a detailed description of the command and all of its parameters, use the man command `man set_power_data`.
`-instance_name` *instance_name*	
	Specifies the hierarchical instance name for which the current files are included. This option is useful when power is calculated for hierarchical partitions of the design, enabling you to read power for the hierarchical partitions For example, `set_power_data -format current -instance_name xy {dynamic_VDD.ptiavg dynamic_VSS.ptiavg}` *Data_type*: string, optional
`-offset` *offset_value*	
	Specifies the start time offset for current waveforms saved inside the current files. The default unit is in seconds; However, you can also specify the unit with the offset time value. This option is useful in the hierarchical flow to align current waveforms of various partitions. It can be also used to shift power-up rush current w UltraSim software. For example, `set_power_data -format current -offset 1ns -instance_name xy {dynamic_VDD.ptiavg dynamic_VSS.ptiavg}` *Data_type*: string, optional
`-power` *value*	

	Specifies the total power of the chip in Watts and is used with -format area option. *Data_type*: float, optional
`-power_output_directory` *dir*	
	Specifies the name of the directory that contains the current files required to run power analysis in parallel with rail analysis. *Data_type*: string, optional
`-repeat_time` *time*	
	Repeats the dynamic current waveform for the specified time. You must ensure that `set_dynamic_rail_simulation -stop` is also specified and duration. This option is only available for current format files during dynamic analysis. The repeat option can be used when multiple current files periods or when IR drop analysis needs to be run for longer duration. See example below. *Default units*: ns. *Data_type*: double, optional
`-repeat_window {`*start end*`}`	
	Specifies to extract the given window from the current waveform and repeat it. You can pick a window such that the current at the `start_time` ar parameter allows you to identify and repeat windows where the start and end time have similar values to ensure that the repeated waveform is no high current values. The unit for the start and end time is ns. The following command specifies that the pattern between {20ns 80ns} will be repeated until simulation stops: `set_power_data -repeat_window {20ns 80ns}` *Data_type*: string, optional
`-reset`	Overwrites previously specified current data. *Data_type*: bool, optional
`-scale` *factor*	
	Specifies a scale factor to apply to the current data. You can use this parameter to scale currents to a more optimistic or pessimistic power-consumption estimate. This parameter is only for `-format current` and `-format ascii`. *Data_type*: float, optional
`-strip_hierarchy_instance_name` *instance*	
	Specifies to use power generated from the top-level rail analysis for block-level rail analysis. This parameter allows you to specify the top-level for which the current files are to be included. When the parameter is specified, the tool removes the prefix from the top-level and reads the top-le during the block-level analysis. The parameter is useful when top-level power data is already available, therefore, you do not need to re-run blo *Data_type*: string, optional

Examples

- The following command specifies a format of type current and a scale of 1 for the three given files.:

```
set_power_data -format current -scale 1 {static_VDD_TDSPCore.ptiavg \ static_VDD_TDSPCore_R.ptiavg static_VSS.ptiavg}
```

- The next example specifies two current data files. The first file specifies vdd_vdd.ptipeak. The second one specifies dynamic_VDD.ptiavg and this current data will be repeated every 100ns, which is the stop time specified for the dynamic analysis.

```
set_power_data -format current vdd_vdd.ptipeak
set_power_data -format current -repeat_time 100ns dynamic_VDD.ptiavg
set_dynamic_rail_simulation -stop 100ns
report_rail ...
```

Related Information

- `set_rail_analysis_config`
- `set_dynamic_rail_simulation`
- `report_rail`
- "Static Power, IRdrop and EM Analysis" in the *Voltus Stylus Common UI User Guide*
- "Dynamic Power and IRDrop Analysis" in the *Voltus Stylus Common UI User Guide*

set_power_pads

```
set_power_pads
-reset | {
[-auto_voltage_source_creation {true|false}]
[-def_shape_only {true | false}]
[-die_instance_name dieinstname]
[-enable_switchnet_ploc_generation {true | false}]
[-format {defpin | padcell | xy | boundary | xyiv}]
[-layer vialayername]
-net net_name
[-number_of_voltage_source value]
[-file file]
[-region {x1 y1 x2 y2}]
[-short_pin_nodes {true | false}]
[-tile {x y}]
}
```

Specifies power pad information.

You can specify multiple power pad formats (`xy` coordinates, pad cell, or power pin) for the same power net, in both, net-based and domain-based rail analysis.

Note: All voltage source names in the power pad location file must be unique for all nets in the domain, e.g. if voltage source name vsrc1 is used in pad location file of net VDD as well as VSS in domain based analysis, program will issue error and exit. In addition, the names of voltage sources in the power pad location file must be different from the names in the DEF pin section.

Parameters

`-auto_voltage_source_creation {true|false}`	
	Specifies to automatically create voltage sources either for all nets (`-net ALL`) or for a specific net (`-net net_name`). The voltage sources are created on the top metal of all via cuts located either on the user-specified via layer or the design top via layer (default). Default: `false`. *Data_type*: enum, required
`-def_shape_only {true | false}`	
	Specifies to generate voltage sources only on the DEF routing shapes. This parameter controls the generation of voltage sources on the cell instance shapes. When this parameter is set to `false`, voltage sources are created for both the DEF routing shapes and shapes inside cell instances. This parameter is specified with the `-auto_voltage_source_creation` parameter. Default: `true` *Data_type*: enum, optional
`-die_instance_name dieinstname`	
	Specifies the die instance name for multi-die analysis. This option is only available in multi-die analysis (`set_rail_analysis_mode -die_mode multi-die`). *Data_type*: string, optional
`-enable_switchnet_ploc_generation {true | false}`	
	Specifies to generate voltage sources on all switched nets. This parameter is specified with the `-auto_voltage_source_creation` parameter. Default: `false` *Data_type*: enum, optional
`-file file`	
	Specifies the name of the file containing the voltage source information. Not required for `-format defpin` option. *Data_type*: string, optional

`-format {defpin | padcell | xy | boundary | xyiv}`		
	Specifies how the voltage source is determined. *Data_type*: enum, required	
	`defpin`	Specifies that voltage sources will be determined by power pins.
	`padcell`	Specifies that voltage sources will be determined by pad cells.
	`xy`	Specifies that voltage sources will be specified by x y coordinates.
	`boundary`	Specifies that voltage source locations will be used for instances in the design.
	`xyiv`	Specifies whether the voltage sources specified in the xy power pad location file are current source or reference voltage source. The format is the same as the xy file, except the additional first and last columns to indicate the current/voltage source and current/voltage value.
`-layer` *vialayername*		
	Specifies the via layer on which the voltage sources are to be generated. If this parameter is not specified, the voltage sources will be generated on the top via layer of the design. This parameter is specified with the `-auto_voltage_source_creation` parameter. *Data_type*: string, optional	
`-net` *net_name*		
	Specifies the name of the power net. *Data_type*: string, required	
`-number_of_voltage_source` *value*		
	Specifies the number of voltage sources to be generated. You can adjust either `-tile` or `-number_of_voltage_source` to reduce the number of voltage sources created. This parameter is specified with the `-auto_voltage_source_creation` parameter. Default: `1 million`. *Data_type*: int, optional	
`-region {`*x1 y1 x2 y2*`}`		
	Generates the voltage sources in the specified region. If this parameter is not specified, the software uses the entire design/layout size as the region. This parameter is specified with the `-auto_voltage_source_creation` parameter. *Data_type*: string, optional	
`-reset`	Resets all the previous `set_power_pads` commands. *Data_type*: bool, optional	
`-short_pin_nodes {true | false}`		
	Specifies to short all interface nodes on a pad cell pin to a single node and create voltage source for that node. Default: `false` **Example**: `set_power_pads -net vdd -format padcell -short_pin_nodes -file pad_cell.file` *Data_type*: enum, optional	
`-tile {`*x y*`}`		

	Defines the number of tiles in the X and Y direction for automatic creation of voltage sources. When `-tile` is specified, the area (`-region` or default design area) will be tiled vertically and horizontally per the given tile size (x*y). For each tile, one voltage source will be created on the top metal of the biggest via array. If `-tile` is not specified, the software creates voltage sources on the top metal of all via cuts located either on the user-specified via layer or the top via layer of the design. You can adjust either `-tile` or `-number_of_voltage_source` to reduce the number of voltage sources created. The `-tile` parameter is specified with the `-auto_voltage_source_creation` parameter. *Data_type*: string, optional

Examples

- The following command specifies the power pad information for the `VDD_TDSPCore_R` net in the file `dtmf_chip.VDD_TDSPCore_R.vsr`c. The voltages sources are specified by x y coordinates.
  ```
  set_power_pads -net VDD_TDSPCore_R -format xy -file \ dtmf_chip.VDD_TDSPCore_R.vsrc
  ```
- The following command specifies `xy`, `padcell`, and `defpin` formats for a single net (`VDD`) as follows:
  ```
  set_power_pads -net VDD -format xy -file VDD.pp
  set_power_pads -net VDD -format padcell -file VDD.padcell
  set_power_pads -net VDD -format defpin
  ```
- The following command specifies power pad information using a combination of user-specified and automatic voltage source creation methods:
  ```
  # Specific setting for net VSS using automatic voltage source creation
  set_power_pads -net VSS -auto_voltage_source_creation true -layer VIA_6

  # Global setting for all nets using automatic voltage source creation
  set_power_pads -auto_voltage_source_creation true -net ALL -layer VIA_5

  # User-specified power pad file for net VDD_AO, therefore automatic voltage source creation feature will not be enabled for this net
  set_power_pads -net VDD_AO -format padcell -file padcell.file -short_pin_nodes true
  ```

Example of power pad location file with XY coordinates for net VDD

```
*-----------------------------------------------------------------------------------------------------------------------------------------------------------
----
* XY power pad location file
* vsrc name X(um) Y(um) LEF/Tech Layer <R=value ohm> <C=value F> <L=value H>
* Package R,L and C parameters are optional
*-----------------------------------------------------------------------------------------------------------------------------------------------------------
-----
VDD100 5.000 793.500 M7
VDD101 1187.835 293.500 M7
VDD102 695.000 5.000 M6
VDD103 1187.835 493.500 M7
VDD104 595.000 1293.800 M6
VDD105 595.000 5.000 M6
```

Example of pad cell file for VDD net,

```
*-----------------------------------------------------------------------------------------------------------------------------------------------------------
----
* Pad cell file for Voltus
* cellname <r=value> <l=value> <c=value>
* Package parameters are optional
*-----------------------------------------------------------------------------------------------------------------------------------------------------------
```

```
-----
PVDD1 r=2.5e-3 l=2.5e-11 c=120e-12
PVDD2 r=2.5e-3 l=2.5e-11 c=120e-12
```

Related Information

- `set_rail_analysis_config`
- `set_net_group`
- "Static Power, IRdrop and EM Analysis" in the *Voltus Stylus Common UI User Guide*
- "Dynamic Power and IRDrop Analysis" in the *Voltus Stylus Common UI User Guide*

set_rail_analysis_config

```
  set_rail_analysis_config
-method {static | dynamic | era_static | era_dynamic}
-accuracy {xd | hd }
-power_grid_libraries dir_list
[-analysis_view view]
[-avg_em_analysis {true | false}]
[-exclude_nets net_name_list]
[-em_models file]
[-em_temperature string ]
[-default_package_resistor value]
[-default_package_inductor value]
[-default_package_capacitor value]
[-voltage_source_search_distance value]
[-report_power_domain {true | false}]
[-tmp_directory_name directory]
[-stream_purpose {metalFill | flipChip | fullChip}]
[-stream_file file [-stream_offset {x y}]]
[-stream_top_cell cell_name ]
[-decap_cell_list { cell1 cell2 ... celln }]
[-filler_cell_list { cell1 cell2 ... celln }]
[-suppress_message {message_id + } ]
[-disable_analysis_types {list of analysis types}]
[-stream_map file]
[-enable_sensitivity_analysis {true | false}]
[-cell_ignore_file filename]
[-write_voltage_waveforms {true | false}]
[-read_thermal_map thermal_map_file]
[-temperature value]
[-include_extraction_tool filename]
[-record_results_start_time time]
[-lef_map lefMappingFile]
[-lifetime value]
[-die_mode {single|multi-die|3dic} -die_instance_name dieinstname [-design designname]]
[-dynamic_trigger_file filename]
[-block_power_up_net_name netname | -powering_up_net net_name1... netname_n]
[-process_techgen_em_rules {true | false}]
[-power_up_sequence_file filename]
[-enable_voltage_source_in_gif {true | false}]
[-work_directory_name directory]
[-power_up_fast_mode {true | false}]
[-ignore_shorts {true | false}]
[-disable_parallel_extraction]
[-enable_distributed_processing_in_solver {true | false}]
[-extraction_tech_file filename]
[-write_hier_block_boundary_voltage {true | false}]
[-check_voltage_source_placement_on_switched_net {true | false}]
[-process_bulk_pins_for_body_bias {true | false}]
[-cluster_via_rule { {via_layer1number_of_equidistant_vias}… }]
[-cluster_via1_ports {true | false}]
[-ignore_fillers {true | false}]
[-ignore_decaps {true | false}]
[-limit_number_of_steps {true | false}]
[-check_thermal_aware_em {true | false}]
[-compress_power_grid_db {true | false}]
[-cluster_via_size value]
[-report_power_in_parallel {true | false}]
[-use_early_view_list filename]
[-use_ir_view_list filename]
[-use_em_view_list filename]
[-enable_mfg_effects {true|false}]
[-enable_rlrp_analysis {true|false}]
[-enable_voltage_across_vias {true|false}]
[-eiv_eval_layers {layerlist}]
[-eiv_eval_window {switching | timing | both | elapse}]
[-eiv_method {worst best avg worstavg bestavg}]
[-reuse_state_directory dir_name]
```

```
[-snap_layer_for_current_taps file_name]
[-enable_scheduler {true | false}]
[-era_current_distribution_factor_for_placed value]
[-era_current_region_file filename]
[-era_current_distribution_layer layer_name]
[-era_current_distribution { unplaced | placed | all | none }]
[-era_add_virtual_via_on_layers  value ]
[-era_lef_layer_map filename]
[-era_power_switch_file filename]
[-era_add_virtual_followpin { standard | extended | none }]
[-era_skip_virtual_via_by_type {whatif | def | all | none}]
[-era_skip_virtual_via_on_layers {{ layer1layer2 } { layer3layer4 } ... }]
[-import_what_if_shapes {true | false}]
[-what_if_shapes_file filename]
[-check_current_balanced_power_grid_em {true | false}]
[-em_threshold value]
[-gif_resolution {low | medium | high}]
[-era_add_virtual_followpin_for_io {true | false}]
[-report_voltage_drop {true | false}]
[-eiv_report{auto | netonly | all}]
[-eiv_thresholdvalue]
[-eiv_max_instancevalue]
[-pre_chain_power_up_sequence_filefilename]
[-rdl_def def_file]
[-rdl_placement{X Y}]
[-rdl_orient{ N|S|W|E|FN|FS|FE|FW }]
[-top_cell_placement{X Y}]
[-top_cell_orient{ N|S|W|E|FN|FS|FE|FW }]
[-watch_location_waveform { {layerNamexCoordyCoord}+ }]
[-enable_rc_analysis{true | false}]
[-em_peak_analysis {true | false}]
[-em_limit_scale_factor {{avg value} {rms value} {peak value}}]
[-em_rms_delta_temperature temp]
[-ict_em_models file]
[-hier_pgv_block_lefs list_of_files]
[-hier_pgv_write_view {ir | em | all}]
[-era_current_distribution_unplaced_area {instance | diearea}]
[-era_current_distribution_nets {net1net2 ...}]
[-era_check_wires_for_generated_current_regions {true | false}]
[-era_tech_lib_generation {true | false}]
[-force_library_merging {true | false}]
[-unconnected_die_package_pins {ignore | error | edit}]
[-mcp_model_mapping {{<mcp model name1> <die DEF name1>} {<mcp model name2> <die DEF name2>} ...}]
[-eiv_pin_based_report {true | false}]
[-gif_zoom_area { x1y1x2y2 }]
[-gif_zoom_top_cell_die_area {true | false}]
[-write_instance_ir_report file]
[-write_instance_pin_ir_report file]
[-report_layer_based_ir_average {true | false}]
[-fine_grain_power_switch_r_on {min avg max}]
[-fine_grain_power_switch_r_on_list filename]
[-ignore_incomplete_net {true | false}]
[-write_combinational_inst_voltage_drop_gif {true | false}]
[-gif_iv_threshold value]
[-verify_logical_connectivity {true | false}]
[-enable_ccs_analysis {true | false}]
[-rlrp_threshold value]
[-rlrp_percentage_threshold value]
[-rlrp_pin_based_report {true | false}]
[-probe_waveform_list filename]
[-probe_pin_voltage_list filename]
[-eiv_eval_nodes {tap | port}]
[-hier_delimiter character]
[-force_extraction  {true | false} ]
[-skip_extraction  {true | false} ]
[-report_power_options option_names]
[-package_trace_connectivity {true | false}]
[-r_effective_pin_report_method { best | worst } ]
[-rlrp_pin_report_method { best | worst | eiv_best | eiv_worst} ]
```

```
[-r_effective_pin_report_layer { top | bottom | all }]
[-rlrp_pin_report_layer { top | bottom | all }]
[-probing_node_file file_name]
[-extend_bump_resolution_in_gif {true | false}]
[-gif_new_color_scale {true | false}]
[-scale_initial_condition_current filename]
[-probe_instance_tap_waveforms {true | false}]
[-rlrp_eval_nodes {port | tap}]
[-report_shorts {true | false}]
[-extract_subconductor_layers {true | false}]
[-eiv_average_per_window_list filename]
[-em_ignore_pgv_resistors {true | false}]
[-eiv_eval_ground_window {true | false}]
[-honor_negative_static_current {true | false}]
[-accumulate_overlapping_pgv_pwl_waveforms {true | false}]
[-report_layers_above_pin_for_instance_ir {true | false}]
[-switching_bit_file_for_eiv_calculationfilename]
[-rlrp_detail_report{true | false}]
[-era_instance_dynamic_current_filefilename]
[-eiv_print_time{true | false}]
[-eiv_histogram_max value]
[-eiv_histogram_min value]
[-eiv_histogram_number_of_bucketsvalue]
[-r_effective_detail_report {true|false}]
[-record_inst_peak_current {true | false}]
[-pre_simulation_period value]
[-pre_simulation_resolution value]
[-fine_pre_simulation_period value]
[-enable_instance_power_gate_report {true | false}]
[-disable_em_split_ac_dc_rules {true | false}]
[-lowest_layer_for_em_check layer_name]
[-ignore_nets_without_voltage_sources {true | false}]
[-eiv_pin_location{true | false}]
[-verbosity{true | false}]
[-enable_reff_analysis {true | false}]
[-rdl_def_list {{rdl_filename1x1y1orient1} {rdl_filename2x2y2orient2} ...}]
[-remove_duplicate_inst {true | false}]
[-write_instance_reff_report file]
[-write_package_pin_current_voltage {true | false}]
[-r_effective_domain_report_format {1 | 2 | 3 | 4}]
[-em_report_line_threshold value]
[-powering_down_net net_name1... netname_n]
[-xpgv_config_file filename]
[-def_based_hierarchical_reports {true | false}]
[-enable_2d_partition_extraction{true | false}]
[-print_gif_range_percentage{true | false}]
[-generate_multi_voltage_library {true | false}]
[-feol_scale_factor non_negative_integer_number]
[-enable_seb {true | false}]
[-env_temperature temperature]
[-seb_lifetime value]
[-seb_table table_filename]
[-seb_temperature temperature]
[-use_rms_delta_t {true | false}]
[-enable_xp {true | false}]
[-xp_cpu_per_job_power value]
[-xp_cpu_per_job_simulation value]
[-xp_host_allocation_method {on_demand | at_startup}]
[-xp_purge {full}]
[-xp_resume {true | false}]
[-xp_reuse_extraction_directory directory_name]
[-xp_simulation_cpu_timeout value]
[-xp_simulation_min_cpu value]
[-enable_package_battery_current_probing{true | false}]
[-probe_package_interface_ports{true | false}]
[-static_multi_mode_analysis {true | false}]
[-em_temperature_layer_list {{<LEF_layer_name> <em_temperature_in_C>}+}]
[-em_rms_delta_temperature_layer_list {{<LEF_layer_name> <delta_T_in_C>}+}]
```

```
[-promote_pin_shapes_only {filler | decap | both} ]
[-unconnected_cell_ignore_filefilename]
[-common_res_inst_pair_file_namefile]
[-flatten_xpgv_block_instances  filename]
[-ignore_duplicate_voltage_source {true | false}]
[-shorting_resistancevalue]
[-tsv_subckt_model_file_list filename]
[-report_instances_missing_current_data {true | false}]
[-ircx_models {RC.ircx EMIR.ircx}|{RC.ircx}]
[-probe_instance_pin_waveforms {true | false}}
[-write_demand_current_region_file filename]
[-em_cdf_percentage value]
[-enable_multi_die_connectivity_check {true | false}]
[-enable_xpgv_scaling { true | false }
[-static_trigger_file filename]
[-rlrp_cell_report_layer filename]
[-peak_em_analysis {true | false}]
[-rms_em_analysis {true | false}]
[-r_effective_eval_nodes {port | tap}]
[-r_effective_report_all {true | false}]
[-set_analyze_bbox {xmin ymin xmax ymax}]
[-override_set_power_data {true | false}]
```

Specifies how the rail analysis will be performed.

Parameters

`-accumulate_overlapping_pgv_pwl_waveforms {true | false}`	
	Specifies to accumulate the current values of Dynamic Detailed View (DDV) PWL waveforms from different trigger events when the PWL waveforms are overlapping with each other. If set to `true`, the current values of the overlapping PWL waveforms are added together for rail analysis. The default value is `false`. *Data_type*: enum, optional
`-accuracy {xd | hd }`	
	Specifies the accuracy mode of analysis. *Default* : `hd` *Data_type*: enum, required
	• `xd`: xd or accelerated definition accuracy mode is used for early implementation stage IR/EM analysis.
	• `hd`: hd or high definition accuracy mode is used for final verification stage IR/EM analysis.
`-analysis_view view`	
	Specifies analysis view created in a CPF (Common Power Format) file. *Data_type*: string, optional
`-avg_em_analysis {true | false}`	
	Specifies to calculate the average EM in the dynamic rail analysis run. This parameter generates the `<net_name>.rj.avg.rpt` report. *Default* : `false` *Data_type*: enum, optional
`-block_power_up_net_name netname`	

	Specifies a net for block power-up analysis. You can specify only one net for block power-up analysis at a time. *Data_type*: string, optional
`-cell_ignore_file` *filename*	
	If specified, excludes all references to the named cells in the specified file, from the design database. *Data_type*: string, optional
`-check_current_balanced_power_grid_em {true | false}`	
	Specifies to check the power-grid EM rules (power-rail terminal-via EM rule). This parameter allows EM comparison with/without this EM rule. This parameter is only applicable in TSMC N20 process nodes where the EM limit may be relaxed if certain constraints are met. Refer to TSMC DRM for more details. *Default* : `false` *Data_type*: enum, optional
`-check_thermal_aware_em {true | false}`	
	Specifies to use the thermal map for EM checks. The thermal map file is specified using the `-read_thermal_map` parameter. The Rail Analysis engine uses the temperature specified in the thermal map file for EM analysis. EM limit has a strong temperature dependency. For a chip that has substantial temperature variation across the chip, the Rail Analysis engine can use the temperature specified in the thermal map file to perform accurate EM analysis. *Default* : `false` *Data_type*: enum, optional
`-check_voltage_source_placement_on_switched_net {true | false}`	
	Specifies to check voltage source placement on switched nets. When this parameter is set to `true`, the software checks whether the specified voltage source location file places any voltage sources on the switched net of the power-gated domain, and disables voltage source placement on the switched net. It also detects shorts and generates an error message. *Default* : `false` *Data_type*: enum, optional
`-cluster_vial_ports {true | false}`	
	Designs with follow-pin routing on M1 and M2 layers have standard cells library LEF with M1, M2, and VIA1 ports. This parameter specifies to cluster VIA1 ports for such cells in order to improve overall extraction and rail analysis performance. *Default* : `true` This parameter is set to `true` by default to improve VIA1 port handling in designs with M1 and M2 follow-pin routing. This parameter gives significantly better performance by clustering equidistant VIA1 ports defined in cell LEF without significant loss of accuracy. However, if you specify a clustering rule, the user specification takes precedence over automated VIA1 clustering. *Data_type*: enum, optional
`-cluster_via_rule { {`*via_layer1 number_of_equidistant_vias*`}...}`	

	Controls the number of vias to cluster on a layer basis. The VIA clustering rule specified using this parameter will override the default clustering rule for a given accuracy mode. The following command will cluster 100 equidistant VIA1 cuts, 200 equidistant VIA2 cuts, and 300 equidistant VIA7 cuts: `set_rail_analysis_config -cluster_via_rule { {VIA1 100} {VIA2 200} {VIA7 300}}` The rest of the VIAs will be clustered using the default clustering rule depending upon the rail analysis accuracy mode. *Data_type*: string, optional
`-cluster_via_size` *value*	
	Specifies the size of the via array to cluster all via layers except VIA1 (layer between Metal1 and Metal2). The default value is `625` (25x25 via array), except when running static analysis using HD mode. The default for static HD mode is `16` (4x4), primarily to preserve more accuracy for EM analysis. *Default* : `625` *Data_type*: int, optional
`-common_res_inst_pair_file_name` *file*	

	Specifies to calculate common resistance along the least resistance path (RLRP) between a pair of cells having the same voltage source. This parameter allows you to check RLRP along cell instance pairs from multiple voltage source locations.
	The format of the specified file is:
	`inst_name1 inst_name2`
	`inst_name3 inst_name4`
	`...`
	The `-common_res_inst_pair_file_name_name` parameter generates a report `<net_name>.common_res` in the following location:
	`<output_dir>/<state_dir>/Reports/<net_name>/`
	The format of the report is:
	`inst_name1 inst_name2 common_res1 pad1`
	`inst_name3 inst_name4 common_res2 pad2`
	`...`
	The following is a snippet from the report:
	`inst_name1 inst_name2` 13.4315 PAD_VDD2

`-compress_power_grid_db {true | false}`	
	Specifies to compress the output files generated by extraction and rail analysis to save disk space usage. This parameter reduces disk space requirement from the extraction and rail analysis stages at approximately 5-10% performance penalty. *Default* : `false` *Data_type*: enum, optional
`-decap_cell_list {` *cell1 cell2 ... celln* `}`	
	Specifies a list of decap cells. This option is used when the decap cells are not tagged in the power-grid library. A cell characterized as a decap cell during power-grid view generation with LibGen does not need to be specified. For dynamic only. *Data_type*: string, optional
`-def_based_hierarchical_reports {true | false}`	

	Specifies to generate separate rail reports for each DEF instance. The net-based effective instance voltage, RLRP, and effective resistance reports are generated hierarchically.
	Reports will be generated for every net in each DEF hierarchy using the following directory structure:
	`<output_dir>/<state_dir>/Reports/HIER/hier_<HIER_ID>/<domain_name | net_name>.<report_type>`
	In the `Reports` folder, the Rail Analysis engine will generate a hierarchical map (`def_hierarchy.map` file) during DEF parsing to associate a hierarchical ID with each DEF instance, as shown below:
	`#hier_id hier_name #def_name`
	`0 chip chipCore`
	Default : `false`
	Data_type: enum, optional

`-default_package_inductor` *value*	
	Specifies the default inductance value of the package resistance-inductance-capacitance (RLC) model to use when you specify voltage source locations. If the power-grid views contain area-based voltage source locations, the following equation is used to define the inductance applied to the voltage sources: `Li = L * number_of_sources` where: `Li` is the inductance on an individual voltage source. `L` is the default inductance set by `-default_package_inductor`. `Number_of_sources` is the number of voltage sources in the power-grid view. The default inductance value is specified in `henrys`. *Data_type*: string, optional
`-default_package_resistor` *value*	
	Specifies the default resistance value of the package resistance-inductance-capacitance (RLC) model to use when you specify voltage source locations. If the power-grid views contain area-based voltage source locations, the following equation is used to define the resistance applied to the voltage sources: `Ri = R * number_of_sources` where: `Ri` is the resistance on an individual voltage source. `R` is the default resistance set by `-default_package_resistor`. `Number_of_sources` is the number of voltage sources in the power-grid view. The default resistance value is specified in `ohms`. *Data_type*: string, optional
`-default_package_capacitor` *value*	

	Specifies the default capacitance value of the package resistance-inductance-capacitance (RLC) model to use when you specify voltage source locations. If the power-grid views contain area-based voltage source locations, the following equation is used to define the capacitance applied to the voltage sources: `Ci = C/number_of_sources` where: `Ci` is the effective capacitance on an individual voltage source. `C` is the default capacitance set by `-default_package_capacitor`. `Number_of_sources` is the number of voltage sources in the power-grid view. The default capacitance value is specified in `farads`. *Data_type*: string, optional
`-design` *designname*	
	Specifies the DEF or GDS design name for multi-die analysis. This option is only available in multi-die analysis (`-die_mode multi-die`). *Data_type*: string, optional
`-die_mode {single|multi-die|3dic}`	
	Specifies whether single or multi-die analysis will be performed. The `3dic` option is used for performing multi-die rlrp analysis. *Default* : `single` *Data_type*: enum, optional
`-die_instance_name` *dieinstname*	
	Specifies the die instance name for multi-die analysis. This option is only available in multi-die analysis (`-die_mode multi-die`). *Data_type*: string, optional
`-disable_analysis_types` { *list of analysis types* }	
	Disables specific analysis types during rail analysis. For example, `set_rail_analysis_config -disable_analysis_types { rlrp vu pi pv }` disables `rlrp, vu, pi`, and `pv` analysis types during `report_rail`. By default, the disable list is empty. The following analysis types can be disabled: `ir tc iv vc dd pi pv javg rj jrms rc node_effr vu`. *Data_type*: string, optional
`-disable_em_split_ac_dc_rules {true | false}`	
	Specifies to use the same AC/DC rules for the EM analysis of the signal and PG nets. When this parameter is set to `true`, the software disables separate handling of the AC/DC rules. By default (when set to `false`), the software uses separate EM rules for the signal and PG nets. *Data_type*: enum, optional
`-disable_parallel_extraction`	
	Specifies to use a single processor for extracting resistance or capacitance. If this parameter is not specified, the software uses parallel processors for extraction. *Data_type*: bool, optional
`-dynamic_trigger_file` *filename*	

	Specifies a dynamic trigger file. When a Macro EM view with current signature is used in a design, it needs to be determine when this block/cell is triggered. The dynamic trigger file is used to determine this. At the trigger time, the dynamic current waveforms of the EM view block will be applied. **Note**: If you specify trigger for cells or instances using this parameter, the tool would give higher precedence to this user-defined trigger file over the auto-generated trigger files in the power analysis directory (`power_output_dir/trigger*.txt`). For more information on the trigger file format, refer to the "Dynamic Trigger File During Rail Analysis" section in the **File Formats** chapter in *Voltus User Guide*. *Data_type*: string, optional
`-eiv_average_per_window_list` *filename*	

	Specifies to generate a report containing different sets of Effective Instance Voltage (EIV) for each timing/switching window for a list of cells/instances.
	
	The format of the specified file is:
	`INST xxx`
	`CELL xxx`
	
	The report contains a set of average EIVs for each instance based on the timing/switching window in the current files. The report file name is: `name.win_avg.rpt`.
	
	Example:
	`set_rail_analysis_config -eiv_average_per_window_list ./clock_network.insts`
	
	File format:
	`INST clk__L2_I1`
	`INST clk__L11_I2`
	`INST clk__L13_I4`
	`INST clk__L13_I3`
	`INST clk__L13_I2`
	
	Output EIV report format:
	`#INST_NAME START_TIME:END_TIME EIV_AVG`
	`clk__L15_I4 1.32e-09:1.78e-09 1.06354 6.38e-09:6.8e-09 1.06217`
	`clk__L15_I3 1.32e-09:1.68e-09 1.06273 6.38e-09:6.72e-09 1.06854`
	`...`
	
	Data_type: string, optional

`-eiv_eval_ground_window {true | false}`	
	Specifies that the ground net switching window should also be considered for the EIV computation. The switching windows from the power and ground nets will be merged into a single switching window. *Default* : `false` *Data_type*: enum, optional
`-eiv_eval_layers {`*layerlist*`}`	
	Specifies to report the EIV of nodes only on the specified layers. When this parameter is not specified, the software reports the EIV values on the worst nodes across all layers. The parameter allows you to analyze the EIV values for the user-defined layers. **Example**: `set_rail_analysis_config -eiv_eval_layers B_M0 -eiv_eval_nodes port` This command reports only the EIV values of nodes on the `B_M0` layer. *Data_type*: string, optional
`-eiv_eval_nodes {tap | port}`	
	Specifies to report EIV based on the instance nodes (tap or port). By default, the software uses the tap nodes for EIV reporting. *Data_type*: enum, optional

`-eiv_eval_window {switching | timing | both | elapse}`	
	Controls the window where EIV is evaluated. EIV is obtained by processing the voltage waveforms and finding the worse-case effective voltage between the power and ground pins during a specific window. There are four possible windows to evaluate EIV: • `switching` – any switching activity (current is non-leakage) reported in power analysis • `timing` – timing window information from TWF or embedded timing engine • `both (Default)` – both switching or timing window • `elapse` – entire rail simulation not filtered to a specific window **Note**: Although power analysis uses timing window information to generate switching activities, it is possible that switching window will be slightly outside timing window if the instance is scheduled to switch at the very beginning or the very end of the timing window. Current waveform may start prior or end later than the time of switching. See "effective instance voltage" for additional information on EIV window in the **Glossary** chapter of *Voltus User Guide*. *Data_type*: enum, optional
`-eiv_histogram_max` *value*	
	Specifies the maximum voltage value of the user-defined voltage range for EIV histogram generation. *Data_type*: float, optional
`-eiv_histogram_min` *value*	
	Specifies the minimum voltage value of the user-defined voltage range for EIV histogram generation. *Data_type*: float, optional
`-eiv_histogram_number_of_buckets` *value*	
	Specifies the number of buckets/ranges to be used for EIV histogram generation. The default number of buckets is 8. The data points are divided into buckets based on the range specified. *Data_type*: int, optional
`-eiv_max_instance` *value*	
	Specifies the maximum number of instances to be included in the .iv files. *Data_type*: int, optional
`-eiv_method {worst best avg worstavg bestavg}`	

	Allows different methods of computing EIV during rail analysis. Using this parameter, you can write out the best, worst, or average effective instance voltage between power and ground nets. The `worstavg` EIV method computes the average instance voltage during each evaluation window defined by the `-eiv_eval_window` parameter, and reports the worst average instance voltage out of all the windows. Similarly, the `bestavg` EIV method computes the average instance voltage during each evaluation window defined by the `-eiv_eval_window` parameter, and reports the best average instance voltage out of all the windows. If `-eiv_method worstavg/bestavg` and `-eiv_eval_window elapse` (i.e. 1 evaluation window) are specified, then this method will report average voltage across all time steps in simulation. *Default*: `worst` See "effective instance voltage (EIV) method" for additional information on EIV methods in the **Glossary** chapter of *Voltus User Guide*. *Data_type*: string, optional
`-eiv_pin_based_report {true | false}`	

	Specifies to report an individual pin's instance voltage for each instance. When enabled, the format of the instance voltage file (.iv file) will be changed to one instance pin per line with pin name added to the last column. If an instance has multiple pins connected to the same net, it will have multiple entries in the .iv file.
	Default : `false`
	The format of the report is:
	<Instance Name> <Instance Voltage> <Cell Name> <Pin Names>
	The following is a snippet of the report from the .iv file:
	`NET`
	`INSTANCESUPPLY 18118 VDD 1.08 POWER`
	`BEGIN`
	`- inst1  1.0798 CKND12 VDD`
	`- inst2  1.0799 INVD6LVT VDD`
	`DOMAIN`
	`INSTANCESUPPLY 18118 VDD 1.08 DIFFERENTIATE`
	`- clk__L13_I3  1.0767 CKND12 VDD VSS`
	`- clk__L13_I4  1.0767 CKND12 VDD VSS`
	Data_type: enum, optional

`-eiv_pin_location {true | false}`	
	Specifies to report an individual pin's location for each instance. This parameter works only with the `-eiv_pin_based_report` parameter. When enabled, the format of the instance voltage file (.iv file) will be changed to one instance pin per line with the pin location added to the last column. This parameter only works for `ivdn` (net-based instance voltage drop). It is not supported for `ivdd` (domain-based instance voltage drop). That is, the pin location will be displayed only in the net-based iv report file (for example, VDD.iv file) and not in the domain-based iv report file (for example, VDD-VSS_div.iv). The default value of the `-eiv_pin_location` parameter is `false`. The format of the report is: *<Instance_Name> <PWR_IV> <Cell_Name> <Pin_Names> <X Y Coordinates>* The following is a snippet of the report from the .iv file: `- INST1 1.13831 MUX2_F8_75LL VDD met1` **`<6152370, 5708640>`** `- INST1 1.13831 ENOR4_F4_75LL VDD met1` **`<6153420, 5714520>`** `- INST2 1.13831 BUF_F1_1SR_75LL VDD met1` **`<6147330, 5708640>`** *Data_type*: enum, optional
`-eiv_print_time {true | false}`	

	Prints the time stamp for the worst voltage in the switching window of the instance, and for the whole (elapse) window. The time stamp values are printed in the detailed EIV report. The default value of this parameter is `false`. Using the time stamp information, you can determine the current signature related to the worst EIV for easy debugging. The default unit for time stamp is `ns`. *Data_type*: enum, optional
`-eiv_report {auto | netonly | all}`	
	Specifies the type of EIV report to be generated. The possible options are: • `auto` - generates domain-based .iv report for domain analysis. • `netonly` - generates net-based .iv report. • `all` - generates both net and domain .iv reports, if available. *Default* : `auto` *Data_type*: enum, optional
`-eiv_threshold` *value*	
	Specifies the EIV drop threshold in mV for reporting. Only voltage drop above the threshold will be saved into the report. *Data_type*: float, optional
`-em_cdf_percentage` *value*	
	Specifies to calculate the Cumulative Distribution Factor (CDF) derating factor based on the EM rules specified in the ICT EM file, given by the `-ict_em_models` parameter. This parameter is used to calculate the fail rate and reliability of a design. The `-em_cdf_percentage` parameter is only applicable to the foundry-specific ICT EM file with the `cdf_percentage` keyword. The value of the CDF percentage should be between 0 and 100. The default value is `1e-9`. The use model of the new parameter is: `set_rail_analysis_config -em_cdf_percentage 5` When the CDF related rule exits in the ICT EM file, the software uses the default value `1e-09` for `-em_cdf_percentage` if it is not specified. *Data_type*: float, optional
`-em_ignore_pgv_resistors {true | false}`	
	Specifies to filter out all the EM violations of resistors inside a macro (resistors coming from PGV). *Default* : `false` *Data_type*: enum, optional
`-em_limit_scale_factor {{avg` *value*`} {rms` *value*`} {peak` *value*`}}`	
	Scales the calculated EM limits by the respective factor for avg, rms or peak current limit specified by `{avg` *value*`} {rms` *value*`} {peak` *value*`}`. The software compares the current through the net against the scaled EM limit to report violations in power EM. *Data_type*: string, optional
`-em_models` *file*	Specifies the name of a file that includes the electromigration (em) models. The `-process_techgen_em_rules` and `-em_models` parameters are mutually exclusive. In the static mode, if `-em_models` is not specified, or if `-process_techgen_em_rules` is not set to `true`, current density (RJ) analysis will be disabled due to lack of EM models. *Data_type*: string, optional
`-em_peak_analysis {true | false}`	

	Specifies to perform peak EM analysis. When this parameter is specified with `-process_techgen_em_rules true` or `-em_models`, both RMS and Peak EM will be reported in the same run. *Default* : `false` *Data_type*: enum, optional
`-em_report_line_threshold` *value*	
	Controls the size of the EM (`rj.avg.rpt`) text report by limiting the number of lines written to the report file. This feature is useful in limiting the report size when you specify a very low EM threshold. **Example**: The following command reports only the top 18 lines: `set_rail_analysis_config -em_report_line_threshold 18` The truncated EM report file has the following note: # NOTE: Aborting EM Report printing as total number of lines has exceeded the Line Threshold "em_report_line_threshold = 18" *Data_type*: int, optional
`-em_rms_delta_temperature` *temp*	
	Specifies the delta temperature for the EM RMS current limit analysis. Default is 5C. *Data_type*: float, optional
`-em_rms_delta_temperature_layer_list {{<`*LEF_layer_name*`> <`*delta_T_in_C*`>}+}`	
	Specifies the delta temperature for specific LEF layers during power EM RMS current limit analysis. *Data_type*: string, optional
`-em_temperature <string>`	Sets a different temperature for EM analysis. As a result, you can use a different but higher t temperature for conservative EM analysis. *Data_type*: string, optional
`-em_temperature_layer_list {{<`*LEF_layer_name*`> <`*em_temperature_in_C*`>}+}`	
	Sets a different temperature for specific LEF layers during power EM AVG current limit analysis. *Data_type*: string, optional
`-em_threshold` *value*	
	Specifies to set EM analysis threshold value for the resistors being reported. Using this parameter, you can control the reporting threshold. By default, rail analysis generates EM report for any resistor having current above `0.9` of its limit. Changing the value to `0` will show every resistor, and `1` will limit to the failed resistors only. *Data_type*: float, optional
`-enable_2d_partition_extraction {true | false}`	

	Specifies to enable the upgraded extraction algorithm that allows 2D partitioning. The 2D extraction algorithm uses an updated capacitance model from the qrcTechFile that provides better accuracy for capacitance calculation. The resistance accuracy level of the 2D extraction algorithm is the same as 1D extraction algorithm. It is recommended to use 2D partitioning as it improves the parallelism of the extraction jobs for large designs. The default value of this parameter is `false`. When the `-enable_2d_partition_extraction` parameter is set to `true`, you must provide the complete qrcTechFile using the `set_rail_analysis_config -extraction_tech_file` parameter. This parameter does not support the 3DIC, inductance extraction, and full-chip GDS flows. *Data_type*: enum, optional
`-enable_ccs_analysis {true | false}`	
	Specifies to enable constant current source analysis. The default value is `false`. *Data_type*: enum, optional
`-enable_distributed_processing_in_solver {true | false}`	
	Specifies to enable distributed processing for Matrix Solver. When set to `true`, Matrix Solver will be able to use multiple hosts across the network to analyze power-grid. *Default* : `true` You can specify this parameter when peak memory of the analysis exceeds machine memory in the local multi-threaded mode. In the distributed mode, peak memory on the local machine will be reduced as child processes will be executed on network hosts. However, the master thread on the local machine will continue to consume majority of peak memory even in the distributed mode. *Data_type*: enum, optional
`-enable_instance_power_gate_report {true | false}`	
	Specifies to generate a report (`powergate.inst.rpt`) that summarizes information about each power gate instance. The default is `false`. The format of the report is: `<inst_name> <cell_name> <Inst X> <Inst Y> <V_Ref> <Ion(total)> <PI(Ipeak/Iavg)> PV(IR)>` Here, • `<inst_name>` is the power gate instance name • `<cell_name>` is the name of the power gate cell • `<Inst X>` is the x coordinate of the power gate instance • `<Inst Y>` is the y coordinate of the power gate instance • `<V_Ref>` is the rail voltage • `<Ion(total)>` is the total current for enabled powergate instance • `<PI(Ipeak/Iavg)>` is the worst Idsat ratio out of all the Ron values • `<PV(IR)>` is the voltage across the power switch using the combination for the worst V(AON Pin)-V(SW Pin) *Data_type*: enum, optional
`-enable_mfg_effects {true | false}`	

	Specifies to honor manufacturing effects during static or dynamic analysis. *Default* : `false` The manufacturing effects are enabled by default in the hd accuracy mode. That is, if you specify `set_rail_analysis_config -accuracy hd`, the `-enable_mfg_effects` parameter is set to `true` by default. *Data_type*: enum, optional
`-enable_multi_die_connectivity_check {true | false}`	
	Specifies to check if voltage sources with `type=nonvsrc` have resistance path to any voltage source with `type=vsrc` inside each die. This parameter allows you to determine if the non-real voltage source (type=`nonvsrc`) is floating or does not have any resistance path to an ideal voltage source (type=`vsrc`), and to ignore floating voltage sources during rail analysis. The "`vsrc`" and "`nonvsrc`" types are given in the .ploc file specified with the `set_power_pads` command. When the `-enable_multi_die_connectivity_check` parameter is specified, the log file displays the list of disconnected terminals for each net. This parameter can be used in both the single-die and multiple-die analysis flows. *Data_type*: enum, optional
`-enable_package_battery_current_probing {true | false}`	
	Specifies to include the battery current waveform data in the `tran_<net>_pad.ptiavg` file. This file includes the voltage of all battery side power_pin and ground_pin mapped pins. *Default* : `false` *Data_type*: enum, optional
`-enable_rc_analysis {true | false}`	
	Enables the Resistor Current (RC) analysis. This parameter supports designs without EM rules/analysis to generate the RC plot for IR drop debugging. *Default* : `false` *Data_type*: enum, optional
`-enable_reff_analysis {true | false}`	

	Specifies to run full-chip effective resistance analysis as part of the static or dynamic IR drop analysis. Effective resistance analysis will be performed on the domain specified with `report_rail`. This parameter allows you to run both resistance and IR drop analyses in a single session.
	The use model of this parameter is:
	`set_rail_analysis_config -enable_reff_analysis  true`
	`report_rail -output_dir <dir_name> -type domain <domain_name>`
	Default : `false`
	Data_type: enum, optional

`-enable_rlrp_analysis {true|false}`	
	Specifies to enable least resistance path analysis for all the instances in the design. The RLRP path can be displayed in the GUI using the `gui_set_power_rail_display -enable_rlrp` command. *Default* : `false` *Data_type*: enum, optional
`-enable_scheduler {true | false}`	

	Controls the Voltus advanced scheduler that improves data handling and multi-CPU scalability of rail analysis . When set to `true`, the scheduler can significantly improve performance and memory consumption, especially when running on large number of processors. The default value is `true`. **Note**: It is recommended to set `-enable_scheduler` to `true`. If set to `false`, some of the advanced extraction features will not be supported. *Data_type*: enum, optional
`-enable_seb {true | false}`	
	Specifies to enable the Statistical ElectroMigration Budgeting (SEB) analysis. *Default* : `false` *Data_type*: enum, optional
`-env_temperature <temperature in °C>`	
	Specifies the ambient temperature used to calculate S_{DC} for SEB analysis. *Default* : `25` *Data_type*: string, optional
`-enable_sensitivity_analysis {true | false}`	
	Only available in the static mode. When set to `true`, the software performs sensitivity analysis on all power-grid segments and highlights those which can be optimized to improve the overall IRdrop profile of the design. The power-grid segments can be optimized using the `scale_what_if_resistance` command. *Default* : `false` *Data_type*: enum, optional
`-enable_voltage_across_vias {true|false}`	
	Specifies to report voltage across vias. *Default* : `false` *Data_type*: enum, optional
`-enable_voltage_source_in_gif {true|false}`	
	Specifies to enable or disable the display of voltage sources in the generated IR drop plot GIFs. When set to `false`, disables voltage sources (power pads) from the *Layout* tab during IR drop analysis. This parameter allows you to control the display of voltage sources in the generated IR drop plot GIFs so that the plot is easily interpreted. *Default* : `true` *Data_type*: enum, optional
`-enable_xp {true | false}`	
	Specifies to run rail analysis in the Extensively Parallel (XP) mode. When this parameter is set to `true`, the rail analysis engine distributes processing over a large number of machines in every stage of the flow (for example, extraction, reporting, GUI, and simulation) to reduce the processing time. *Default* : `false` *Data_type*: enum, optional
`-enable_xpgv_scaling { true | false }`	

	Specifies to enable xPGV current scaling that is calculated as a ratio of voltages. If PGV voltage is V1 and analysis voltage is V2, then the scaling factor would be V2/V1. This factor is applied to each power net. In case there are multiple power nets and a single ground net, the tool will use the worst scaling factor from the power rails and apply it to the ground net. This parameter is used for scaling of currents stored in xPGV. The parameter allows you to use the same xPGV in different designs, irrespective of the voltage at which the xPGV is generated. `-enable_xpgv_scaling` and `-static_trigger_file` are mutually exclusive. *Default* : `false` *Data_type*: enum, optional
`-extend_bump_resolution_in_gif {true | false}`	
	Specifies to increase the size of voltage sources represented for the voltage source current (vc) and voltage drop across package (vu) plots to improve visibility. *Default* : `false` *Data_type*: enum, optional
`-era_check_wires_for_generated_current_regions {true | false}`	
	Specifies to generate unplaced current regions only in areas where actual net wires are present. The default value is `false`. The current regions are generated from the power domain information present in the design. *Data_type*: enum, optional
`-era_current_distribution_factor_for_placed` *value*	
	When you use `-era_current_distribution` and set to distribute current for placed blocks/macros, the ratio of power allocated to each placed instance is calculated based on its total area. Using this parameter, you can control the current distribution factors for the placed blocks/macros, hence power allocated for area-based power calculation. For example, if you specify 0.5, the tool will assume all placed blocks/macros to be 50% of its actual size and distribute current accordingly. *Default* : `1` *Data_type*: float, optional
`-era_current_region_file` *filename*	

	Specifies a file that includes a list of regions and the amount of current to be distributed within them for the power-grid. Default units in file are `mA`. You can also specify rectilinear current regions in the file.
	#Format:
	`LABEL name NET netName AREA x1 y1 x2 y2 LAYER layername <CURRENT value | PWL (t1 i1 t2 i2 …)> INTRINSIC_CAP <value> LOADING_CAP <value> [ADD_OVERLAP | SUBTRACT_OVERLAP]`
	#Unit:
	current mA, cap pf, time ns, coordinate um
	• `ADD_OVERLAP`: Honors all overlapping regions, and adds current in the overlapped regions.
	• `SUBTRACT_OVERLAP`: Specifies that if two XY regions overlap, the first specified region will be honored, and the subsequent region will subtract the overlapped area to apply the region current to the remainder area.
	Static Example :
	`label test1 net VDD area 100 200 400 500 layer M1 current 10`
	Dynamic Example :
	`label test1 net VDD area 100 200 400 500 layer M1 pwl (0ns 0mA 1ns 0mA 1.9ns 0mA 2ns 10mA) intrinsic_cap 10 loading_cap 60`
	Data_type: string, optional

`-era_current_distribution_layer layer_name`	
	Specifies the layer name for distributing unplaced current in the early rail analysis mode. *Data_type*: string, optional
`-era_current_distribution { unplaced | placed | all | none }`	
	This parameter controls the behavior of era current distribution. `set_power_data` area based power, or `current_region_file` need to be specified for ERA current distribution to work. Placed instanced without uti or ascii based power can also be considered for era current distribution. • Unplaced: Enable current distribution only for unplaced instances. If `set_power_data` area based power is specified, `-era_current_distribution_layer` will be required. • Placed: Enable current distribution for placed instances without any power specified. • All: Both unplaced and placed instances without power specified; will have ERA current. • None: Disable ERA current distribution. *Data_type*: enum, optional
`-era_current_distribution_nets {net1 net2 ...}`	
	Specifies to distribute current for unplaced instances by total die area on specified nets only. The default behavior is to distribute current for all nets. This parameter will be required when `-era_current_distribution_unplaced_area diearea` is used. For single net analysis, you can either specify a single net (for example, `-era_current_distribution_nets {VDD_AO}`) or perform the analysis without this parameter. *Data_type*: string, optional
`-era_current_distribution_unplaced_area {instance | diearea}`	
	Specifies to distribute current for unplaced instances by total die area or actual instance area. The default value is `instance`. When `diearea` is specified, the entire unplaced die area is considered for computation of current distribution. The die area considered excludes user-specified current regions, placed instances, and placement blockages. *Data_type*: enum, optional

`-era_add_virtual_followpin { standard | extended | none }`	
	Specifies to generate virtual followpins. The `extended` followpins will create followpins that extend from one stripe to another. The `standard` followpins may extend to previous stripe but does not reach the next stripe. *Default*: `none` *Data_type*: enum, optional
`-era_add_virtual_followpin_for_io {true | false}`	
	By default, ERA virtual follow pin insertion only works in the core area. When this parameter is set to `true`, ERA follow pin routing is extended to the IO area as well. *Data_type*: enum, optional
`-era_add_virtual_via_on_layers` *value*	
	Enables you to control on which layers virtual via can be inserted in early rail analysis. Using this parameter, virtual via can only be inserted between each pair of layer names provided in the design. Alternatively, you can provide wildcard, '*', as the second layer name and virtual via will be allowed to be inserted from the first layer to any layers in the design. For example, `-era_add_virtual_via_on_layers { {M1 *} }` will allow virtual via to be inserted from M1 to any layers in the design. This parameter does not have a default value but ERA will insert virtual via for all layers if -era_skip_virtual_via parameters are not provided. *Data_type*: string, optional
`-era_instance_dynamic_current_file` *filename*	

	Specifies an instance/cell based PWL file in the dynamic ERA flow (`set_rail_analysis_config -era_dynamic`). This feature allows you to perform dynamic ERA without having to specify the dynamic .ptiavg/.ptiavk current files (`set_power_data`).
	
	The specified PWL should cover the entire simulation period. If the simulation period is longer than the specified PWL, the software will assume 0 current for the remaining period (`set_dynamic_rail_simulation -stop <time>`). You can also use the `set_dynamic_rail_simulation -auto_repeat` parameter to repeat the dynamic current waveform till the `-stop` time specified, instead of assuming 0 current for the remaining period.
	
	The format of the file is:
	`INSTANCE | CELL <NAME> PWL { <time> <current> } pwrnet|gndnet <net_name> [ADD_OVERLAP|SUBTRACT_OVERLAP]`
	
	• Default time unit = `ns`.
	• Default current unit = `mA`
	• `pwrnet|gndnet` keyword indicates if the specified net is a power or ground net.
	• `ADD_OVERLAP` keyword indicates to honor all overlapping instance currents, that is, instance to instance overlap and instance to region overlap.
	• `SUBTRACT_OVERLAP` keyword indicates to honor instance current, and subtract its area from the overlapped X-Y region to apply the X-Y region current to the remainder area. `SUBTRACT_OVERLAP` is default. This keyword is not applicable for instance to instance overlap.
	
	The following is a snippet of the specified file:
	`INSTANCE top/instA PWL { 0ns 0mA 0.1ns 1mA 0.2ns 0.5mA } pwrnet VDD`
	`INSTANCE top/instA PWL { 0ns 0mA 0.1ns -1mA 0.2ns -0.5mA } gndnet VSS`
	`CELL macro_A { 0 0 0.1ns 1 0.2 0.5 }`
	`...`
	
	If you have specified both the `-era_instance_dynamic_current_file` and `-era_current_region_file` parameters, and there are overlapping instances with the specified current region, the `SUBTRACT_OVERLAP` behavior is implemented by default.
	
	Data_type: string, optional

`-era_lef_layer_map` *filename*	

	If a technology library or an extraction techfile is not provided, ERA can automatically create a technology library and the layermap needed to generate the library. If the automatic layer map generation fails, you can use this parameter to manually specify a layer map file.
	
	The format of the layer map file is:
	
	File Format:
	`Layer Type tech_layername lefdef lefdef_layername`
	`metal METAL_1 lefdef M1`
	`via VIA_1 lefdef VIA1`
	`metal METAL_2 lefdef M2`
	`via VIA_2 lefdef VIA2`
	`metal METAL_3 lefdef M3`
	
	Data_type: string, optional

`-era_power_switch_file` *filename*	

	If a power gate PGV is not provided, this parameter specifies the name of the power gate (power switch) file that will be used to perform power gate steady-state analysis.
	
	File format:
	
	`CELL <CELLNAME> SUPPLY <SUPPLY_PIN> SWITCHED <SWITCHNET_PIN> RON <VALUE> IDSAT <VALUE> ILEAK <VALUE>`
	
	`CELL HDRSID0 SUPPLY TVDD SWITCHED VDD RON 500 IDSAT 1 ILEAK 0.001`
	
	Data_type: string, optional

`-era_skip_virtual_via_by_type {whatif | def | all | none}`	

	Specifies to skip a given via type. By default, ERA generates all virtual via layer types. • `whatif` vias are virtual vias that have connectivity to user-defined what if shapes. • `def` vias are virtual vias between two metal shapes defined in DEF. • `all` will skip all virtual via generation. • `none` will insert both what-if and DEF vias. It will insert vias on all layers, unless the layers are controlled by other parameters. *Default*: `all` *Data_type*: enum, optional
`-era_skip_virtual_via_on_layers {{ layer1 layer2 } { layer3 layer4 } ... }`	
	Skips via insertion between stripes and non-stripes, on the specified LEF layer pairs. *Data_type*: string, optional
	{ layer1 layer2 } { layer3 layer4 } ...: Specifies the LEF layer names.
`-era_tech_lib_generation {true | false}`	
	Specifies to skip technology PGV generation in the early rail analysis mode. This parameter skips technology PGV generation even if the `-extraction_tech_file` parameter is specified. The default value is `true`. *Data_type*: enum, optional
`-extract_subconductor_layers {true | false}`	
	Specifies to perform extraction of the sub-conductor layers. The default is `false`. *Data_type*: enum, optional
`-extraction_tech_file filename`	
	Specifies the extraction technology file to be used for top-level power-grid extraction. If you do not specify this parameter, the software will use the extraction technology file stored inside the power-grid view library. *Data_type*: string, optional
`-feol_scale_factor non_negative_integer_number`	
	Specifies to scale the front-end-of-line (FEOL) self-heat value for all instances in the self-heating analysis flow. The delta T FEOL will be calculated based on the static power, thermal resistance, and FEOL scale factor. *Data_type*: float, optional
`-filler_cell_list { cell1 cell2 ... celln }`	
	Specifies a list of filler cells. This option is used when the filler cells are not tagged in the power-grid library. *Data_type*: string, optional
`-fine_grain_power_switch_r_on {min avg max}`	
	Specifies whether the minimum, maximum, or average Ron will be used for a powergate cell in finegrain memory characterization with multiple Ron values. The default value is `max`. *Data_type*: enum, optional
`-fine_grain_power_switch_r_on_list filename`	

	Specifies the name of a file containing the powergate cell names and their corresponding Ron values (minimum, maximum, or average) to be used in finegrain memory characterization with multiple Ron values. The format of the file is:
	`CELL cell1 min/max/avg`
	`CELL cell2 min/max/avg`
	`INST inst1 min/max/avg`
	Note: The `-fine_grain_power_switch_r_on` and `-fine_grain_power_switch_r_on_list` parameters allow you to specify multiple Ron values in a scenario where the Ron value can change with respect to different modes of simulation.
	Data_type: string, optional

`-fine_pre_simulation_period` *value*	
	Specifies the fine-grain pre-simulation period that rail analysis will use. The supported time units are s, ms, us, ns, and ps. If no explicit time unit is specified, the default ns (nanosecond) will be applied. The time step in the fine-grain pre-simulation period will be the same as the simulation period. Pre-simulation allows you to determine the current profile and analyze IRdrop of the circuit when power supply is in steady state. **Note**: When both the coarse-grain (`-pre_simulation_period`) and fine-grain pre-simulation periods are specified, the fine-grain pre-simulation period will start after the coarse-grain pre-simulation period. The simulation phase starts after the fine-grain pre-simulation period. Both fine-grain pre-simulation and coarse-grain pre-simulation can be specified just by themselves without requiring the other one to be specified. In the case where fine-grain pre-simulation is not specified, the simulation will start right after the coarse-grain period. *Data_type*: float, optional
`-flatten_xpgv_block_instances` *filename*	

	Specifies a file that includes a list of block instances that use DEF instead of xPGV for rail analysis, that is, specify the instances that are not to be used as xPGV. This parameter allows you to specify a few instances of the same block that will use DEF, and the others that will use xPGV at the top-level. If you do not specify this parameter, and if both DEF and xPGV are given for a block, xPGV takes precedence over DEF.
	Instance List File Format:
	`<block_name>  <block_instance_name|All|None>`
	In this file, you can specify if a block is using a DEF by specifying the block instance name, or with the `All` or `None` keywords. `All` indicates that all instances are using DEF, and `None` indicates that none of the instances are using DEF.
	Example:
	`CORE  /top/x1/x12/instA`
	For a design with three blocks, if you want to use the DEF and xPGV as following:
	Block_1 - > A1 (xPGV) & A2 (DEF)
	Block_2 - > B1 & B2 (xPGV for both instances)
	Block_3 - > C1 & C2 (DEF for both instances)
	The specified file will contain the following information:
	`Block_1 A2`
	`Block_2 None`
	`Block_3 All`

`-force_extraction {true | false}`	
	Specifies whether the existing work directory extracted data should be used, or the design needs to be re-extracted during rail analysis. *Default* : `false` *Data_type*: enum, optional
`-force_library_merging {true | false}`	

	Specifies to force merge PGVs with different resistivity and layers during rail analysis, wherein the first library definition will be used. *Default* : `false` *Data_type*: enum, optional
`-generate_multi_voltage_library {true | false}`	

	When this parameter is specified, all the PGVs for each cell will be considered, and voltage-dependent capacitance information will be available for use in rail analysis, similar to the multi-voltage PGVs.
	The `-generate_multi_voltage_library` parameter is used in a scenario when you have generated multiple PGVs for different voltages, instead of generating a single multi-voltage PGV that has voltage-dependent capacitance. The default value of the parameter is `false`.
	The following example shows the voltage-dependent capacitance table for the cell INV:
	==========
	`CellName Net Name Voltage Capacitance`
	`INV VDD 1.0 1.1e-15`
	`INV VDD 0.9 1.0e-15`
	Here, there are two characterization voltage values and their corresponding capacitance values. The rail analysis engine uses both the voltage-dependent capacitance values to perform interpolation based on the two voltage values.
	Data_type: enum, optional

`-gif_iv_threshold` *value*	
	Specifies the instance voltage threshold for the combined instance voltage drop GIF for all the nets specified in the domain. The threshold is specified in voltage drop value. Any instance voltage drop value greater than the threshold will be put into the red bucket range in GIF, and the rest of the range will be linearly distributed down to the minimum instance voltage drop. *Data_type*: float, optional
`-gif_new_color_scale {true | false}`	
	Specifies to change the color scheme of the GIF file generated in the state directory during the static and dynamic rail analysis flow to match the GUI color scale. *Data_type*: enum, optional
`-gif_resolution {low | medium | high}`	
	Controls the resolution of the generated GIFs after rail analysis. This option can have the following values: `low`, `medium`, and `high`. The default value is `medium`. *Data_type*: enum, optional
`-gif_zoom_area { x1 y1 x2 y2 }`	
	Specifies to set the GIF generation bounding box to the given area (in microns). *Data_type*: string, optional
`-gif_zoom_top_cell_die_area {true | false}`	
	Specifies to set the GIF generation bounding box to the topcell DIEAREA listed in the DEF file, instead of the total design size including RDL. If RDL is not used, this parameter has no effect. This parameter will change the bounding box of all the GIFs generated in rail analysis. **Note:** The `-gif_zoom_area` and `-gif_zoom_top_cell_die_area` parameters are mutually exclusive. If both the parameters are specified, the `-gif_zoom_area` parameter takes precedence. *Data_type*: enum, optional
`-hier_delimiter` *character*	

	Specifies the hierarchical delimiter in the DEF file. The default hierarchical delimiter is a forward slash (/) but can be changed by setting the `-hier_delimiter` parameter. You can use this parameter to override the existing hierarchical delimiter. *Data_type*: string, optional
`-hier_pgv_block_lefs` *list_of_files*	
	Specifies the block LEF file for all hierarchical DEF blocks in the design. This will trigger automatic hierarchical PGV generation for the design hierarchy. The software will error out if any of the LEF files are missing for a DEF hierarchy. *Data_type*: string, optional
`-hier_pgv_write_view {ir | em | all}`	
	Specifies the view type to generate for the hierarchical PGV. The allowed values are "`ir`", "`em`" and "`all`". "`all`" means both `ir` and `em` views will be created. If you choose to only generate `ir` or `em` view, disk consumption can be reduced since only one view will be generated. *Default*: `all` *Data_type*: enum, optional
`-honor_negative_static_current {true | false}`	
	Specifies to honor negative current value in the current file (`.ptiavg`) during static rail analysis. The default is `false`. *Data_type*: enum, optional
`-ict_em_models` *file*	
	Specifies the ICT-EM file which only contains the EM rules and excludes the RC extraction data in ICT. The software will use the EM rules in the specified ICT-EM file to calculate EM limits for power net EM analysis even if there are embedded EM rules in the Quantus tech file. *Data_type*: string, optional
`-import_what_if_shapes {true | false}`	
	Specifies to import the what-if shapes created by the create_what_if_shape command for rail analysis. **Note**: If both the `-import_what_if_shapes` and `-what_if_shapes_file` parameters are specified, union of two sets of what-if shapes is considered for analysis. *Data_type*: enum, optional
`-include_extraction_tool` *filename*	
	Specifies extractor (ZX) commands and variables in the form of an include file. *Data_type*: string, optional
`-ignore_decaps {true | false}`	
	Ignores decap cells during rail analysis. It ignores cells that are tagged as DECAP cells during library characterization or set as decap cells using the `set_rail_analysis_config -decap_cell_list` command parameter. If decaps are used to preserve connectivity on the follow-pin routing, you must set this parameter to `false`. Ignoring decaps during dynamic analysis is not recommended. *Default*: `true` for static, `false` for dynamic *Data_type*: enum, optional
`-ignore_duplicate_voltage_source {true | false}`	

	Specifies to check for multiple voltage sources with the same name in the power pad (.ploc) file. When set to `true`, the duplicate voltage sources are ignored. When this parameter is set to `false` and duplicate voltage sources are present in the .ploc file, an error will be generated. If there are duplicate voltage sources, rail analysis will report an error message like the following: `** ERROR: (VOLTUS_RAIL-3647): Failed to run rail analysis due to redefined voltage source name in power pin location file(s) - /vsrc_same_name/PADs/VDD_AO.pp.` *Default*: `true` *Data_type*: enum, optional
`-ignore_fillers {true | false}`	
	Ignores filler cells during rail analysis if the list of filler cells are defined in PGV generation, or specified as fillers using the `set_rail_analysis_config -filler_cell_list` command parameter. During rail analysis, ignore library cells that are tagged as FILLER cells and have no capacitance associated with the interface nodes. If fillers are used to preserve connectivity on the follow-pin routing, you must set this parameter to `false`. *Default*: `true` *Data_type*: enum, optional
`-ignore_incomplete_net {true | false}`	
	Specifies to skip power and ground nets that are not defined in the design, or have been defined but with no instance connection. *Data_type*: enum, optional
`-ignore_nets_without_voltage_sources {true | false}`	
	Specifies if Rail Analysis should continue or exit with an error message if there are nets with no voltage sources. The possible arguments are: • `true`: prints an error message only but will allow the Rail Analysis run to continue. • `false`: exits immediately and returns to the Tcl prompt. If all the VDD/VSS nets have no voltage sources attached, the Rail Analysis run will error out and exit even if `-ignore_nets_without_voltage_sources` is set to `true`. *Default* : `false` *Data_type*: enum, optional
`-ignore_shorts {true | false}`	
	This parameter allows you to ignore shorts that are found during rail extraction and continue rail analysis. However, `-ignore_shorts` cannot be applied if the design contains GDS. *Data_type*: enum, optional
`-ircx_models {RC.ircx EMIR.ircx}|{RC.ircx}`	
	Specifies an encrypted or unencrypted RC/EM technology (.ircx) file. The foundry-specific IRCX file, which contains the process information for extraction and EM modeling, can be used when the ICT-EM file is not available for electromigration analysis. You can either specify two IRCX files, one file containing the RC process data (`RC.ircx`) and the other file containing EM rules (`EM.ircx`), or a single IRCX file that includes both the RC and EM sections (`RC.ircx`). A single IRCX file is used for the mature nodes, and separate RC and EM IRCX files are used for the advanced process nodes. *Data_type*: string, optional
`-lef_map` *lefMappingFile*	

	Specifies the LEF layermap file, which contains the mapping of the LEF layer and the technology layer. The LEF layermap file format is as follows: `layer map lef_layer_name tech_layer_name` *Data_type*: enum, optional
`-lifetime` *value*	
	Specifies the lifetime value in hours. *Default* : 87,600 hours (= 10 years) *Data_type*: int, optional
`-limit_number_of_steps {true | false}`	
	Specifies to limit the number of steps (simulation period, and resolution or step size) during dynamic IRdrop analysis or power-up analysis. The default value of this parameter is `true`. If the number of steps during dynamic IRdrop analysis or power-up analysis exceeds `1000`, the tool gives an error message and exits analysis. This parameter ensures that you are aware that there could be a problem in how the analysis is set up. If the performance risks are understood, you can set this parameter to `false` and re-run the analysis. The step-size and simulation period for the analysis is derived from current files generated by power analysis. You can explicitly set the step size and simulation period during power analysis using the `set_dynamic_power_simulation` command. During power-up analysis, the simulation period of the analysis is derived based on the turn-on time of the last power-switch in the power-up domain. If this time is too large, you may want to check if the turn-on times of the power-gates are correctly captured in the timing window file. Optionally, you can reduce the number of steps by explicitly specifying the stop time during rail analysis using the `set_dynamic_rail_simulation` command. *Data_type*: enum, optional
`-lowest_layer_for_em_check` *layer_name*	
	Specifies the layers that should not be displayed in the EM plots (such as, rj, jrms, and javg) and reports. *<layer_name>* is the layer below which the EM plots and reports will not be generated. The default value is `false`. *Data_type*: string, optional
`-mcp_model_mapping {{`*<mcp model name1> <die DEF name1>*`} {`*<mcp model name2> <die DEF name2>*`} ...}`	
	Specifies to set the mapping between the package model and die name. *Data_type*: string, optional
`-method {static | dynamic | era_static | era_dynamic}`	
	Specifies whether static or dynamic rail analysis, or static or dynamic early rail analysis will be performed. *Default* : `static` *Data_type*: enum, required
`-exclude_nets` *net_name_list*	
	Specifies a list of nets to exclude from analysis. Used in designs with power switches. *Data_type*: string, optional
`-override_set_power_data {true | false}`	

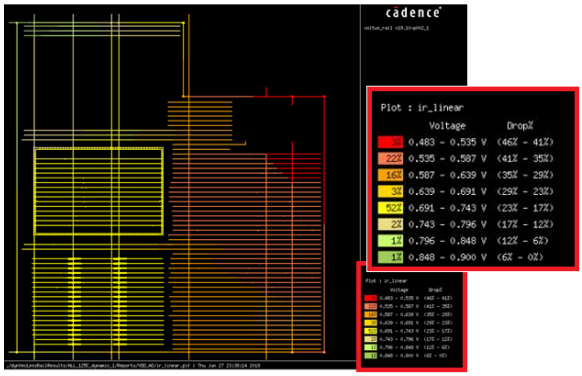
	Default: `false` Specifies to override the previous settings of the `set_power_data` command and use the last specified setting of `set_power_data` when targeting the same design object using the same current file. This feature is convenient when many `set_power_data` commands have been set and you want to override one of them with a new scaling factor or other options. **Example**: In the following example, the first command will be discarded and the second command will be honored: • `set_power_data -format current -instance_name XYZ dynamic_VDD.ptiavg` • `set_power_data -format current -instance_name XYZ dynamic_VDD.ptiavg -scale 2.0` *Data_type*: enum, optional
`-package_trace_connectivity {true | false}`	
	When `-package_trace_connectivity` is set to `true`, connectivity tracing is enabled from the die pins to the board pins of all the elements in the package model file. When this parameter is set to `false`, connectivity tracing is disabled. *Default* : `false` *Data_type*: enum, optional
`-peak_em_analysis {true | false}`	
	Specifies to calculate the peak EM in the dynamic rail analysis run. This parameter generates the `<net_name>.rj.peak.rpt` report. *Default* : `false` *Data_type*: enum, optional
`-power_grid_libraries` *dir_list*	
	Specifies the name of the power-grid view library directories. *Data_type*: string, optional
`-power_up_fast_mode {true | false}`	
	Specifies to enable fast mode simulation for native power-up analysis. When the fast mode is enabled, the software uses lumped parasitic model for all instances connected to the powering up net. The fast mode native power-up analysis allows you to quickly estimate turn-on time and rush current injected in the global power-grid as the power-gated net is powering up. The fast-mode power-up analysis gives faster turn-around time and similar accuracy as accurate power-up analysis. *Default* : `true` *Data_type*: enum, optional
`-power_up_sequence_file` *filename*	

	Specifies the name of the file containing the firing time for each power gate instance. The power-up sequence file format is as follows:
	`<power gate instance name> <pin name> <always- on net> <switched net> <rise_time> <rise_slew> <fall_time> <fall_slew> <control pin voltage>`
	`<control pin voltage>` is optional.
	A sample power-up sequence file is given below:
	`inst_name E VCC_NORMAL VCC_SWITCHED 5.50E-09 2.90E-10 5.50E-09 2.90E-10 0.84`
	`inst_name E VCC_NORMAL VCC_SWITCHED 5.93E-09 7.23E-10 5.93E-09 7.23E-10 0.84`
	Data_type: string, optional

`-powering_up_net` *net_name1* ... *netname_n*	
	Specifies a list of powering up nets. *Data_type*: string, optional
`-powering_down_net` *net_name1... netname_n*	
	Specifies a list of powering down nets. *Data_type*: string, optional
`-pre_chain_power_up_sequence_file` *filename*	

	Specifies a power-up sequence file that contains the pre-chain start/end and mother (data) chain start/end timing information. In the power-up flow, you can specify this pre-chain sequence file instead of a complete power-up sequence file to modify the ramp-up timing of different power switches using the timing constraint in the specified file. The timing information for the other power-gate instances is calculated using the pre-chain sequence file.
	The format of the file is:
	`Prechain_start <pggate_instance_name> <pin_name> <d_time> <d_delay> <d_slew>`
	`Prechain_end <pggate_instance_name>`
	`Datachain_start <pgagate_instance_name> <pin_name><m_time> <m_delay> <m_slew>`
	`Datachain_end pggate_instance_name`
	Data_type: string, optional

`-pre_simulation_period` *value*	
	Specifies the pre-simulation period that rail analysis will use. The supported time units are s, ms, us, ns, and ps. If no explicit time unit is specified, the default ns (nanosecond) will be applied. The pre-simulation period include the coarse-grain and fine-grain pre-simulation periods. If `-fine_pre_simulation_period` is not specified, then the entire pre-simulation period will do coarse pre-simulation. *Data_type*: float, optional
`-pre_simulation_resolution` *value*	
	Specifies the transient time step size or resolution. The supported time units are s, ms, us, ns, and ps. If no explicit time unit is specified, the default ps (picosecond) will be applied. *Data_type*: float, optional
`-print_gif_range_percentage {true | false}`	

	Specifies to display the plot title and voltage drop percentage (`Drop%`) for the following GIF files:
	• **IR** (File Location: `<rail_dir>/Reports/<power_net>/`)
	`ir_linear.gif`
	`ir_limit.gif`
	• **EIV** (File Location: `<rail_dir>/Reports/`)
	`<pg_pair>.<eiv_method>_eiv.gif`
	`<pg_pair>.<eiv_method>_limit_eiv.gif`
	
	`IR Drop%` is calculated by 1-(voltage/ideal_voltage), and `EIV Drop%` is calculated by 1-(effective_voltage/ideal_effective_voltage)
	The following image shows the voltage drop percentage for the `ir_linear` plot:
	 
	Default : `false`
	Data_type: enum, optional

`-probe_instance_pin_waveforms {true | false}`	
	Specifies to probe current and voltage waveforms for all the pins belonging to probing instances. When `-probe_instance_pin_waveforms` is set to `true`, it writes all the voltages and currents of the pins of probing instances to the waveform files. The use model of this parameter is: `set_rail_analysis_config -method dynamic -probe_instance_pin_waveforms true` The watched instance has the pin names in the waveform file, as shown below: Voltage waveform name: `<instance_name>:ir` Current waveform name: `TC_<instance_name>:ir` *Default* : `false` *Data_type*: enum, optional
`-probe_instance_tap_waveforms {true | false}`	

	Specifies to probe current and voltage waveforms for all tap nodes belonging to probing instances. When `-probe_instance_tap_waveforms` is set to `true`, it writes all the tap node voltages and currents of the instances from `-probe_waveform_list` to the waveform files. The default value of the new parameter is `false`. The use model of this parameter is: `set_rail_analysis_config -method dynamic -probe_instance_tap_waveforms true -probe_waveform_list` *`instance_list_file`* The instance's tap nodes are named based on instance name, and its node ID and location in the waveform file, as shown below: Voltage waveform name: `<instance_name>:N_...:X_..:Y_:ir` Current waveform name: `TC_<instance_name>:N_...:X_..:Y_...` *Data_type*: enum, optional
`-probe_package_interface_ports {true | false}`	
	Specifies to probe voltages for all the ports of the top-level subcircuit and the interfaces within the package. When this parameter is specified, the rail analysis engine retrieves the package interface ports from the package model and writes them into a probing node file to watch their voltage waveforms. *Default* : `false` *Data_type*: enum, optional
`-probe_pin_voltage_list` *`filename`*	
	Writes the voltage of the interface nodes for the specified instance/cell to a block boundary voltage file. The software generates a block boundary voltage file with file name based on the block instance name, `<instance_name>.pp` for every instance/cell in the list. *Data_type*: string, optional
`-probe_waveform_list` *`filename`*	

	Specifies the name of a file containing the instance and cell name(s) for which you want to capture voltage and current waveforms. This parameter allows you to control whether to probe voltage or current waveforms or both.
	You can specify the optional parameter `PROBE <VOLTAGE|CURRENT|ALL>` in the file. Default is `ALL`. When `VOLTAGE` or `CURRENT` is specified, only the voltage or current waveforms will be probed.
	The format of the file is:
	`CELL/INST <`*`name`*`>`
	Following is an example of the file:
	`INST I1_1`
	`INST I2_1`
	Data_type: string, optional

`-probing_node_file` *`file_name`*	

	Specifies the probe node file containing the probe point on a die. The file format for specifying probing nodes is as follows:
	`<user defined unique probe name> <location = x> <location = y> <layer> <Power/ground net name>`
	You can also specify the package node voltages and resistor currents that are to be probed in the following format:
	`<probe name> pkg ir <pkg circuit node name>`
	`<probe name> pkg rc <pkg circuit resistor name>`
	Example
	`probe_1 18 16 METAL_1 VDD`
	`# pkg node voltage`
	`probe_pkg_1 pkg ir GND.in0`
	`probe_pkg_6 pkg ir VDD.in2`
	`# pkg resistor current`
	`probe_pkg_cur_1 pkg rc R_1_1`
	`probe_pkg_cur_2 pkg rc R_5_2`
	Note: The units of x and y should be in microns.
	Data_type: string, optional

`-process_techgen_em_rules {true | false}`	
	Processes the EM rules defined in the extraction technology file. When `-process_techgen_em_rules` is specified with `-extraction_tech_file`, the extraction technology file will be processed. If `-process_techgen_em_rules` is specified without `-extraction_tech_file`, the .cl model will be processed. *Default* : `false` The `-process_techgen_em_rules` and `-em_models` parameters are mutually exclusive. In the static mode, if `-em_models` is not specified, or if `-process_techgen_em_rules` is not set to `true`, current density (RJ) analysis will be disabled due to lack of EM models. *Data_type*: enum, optional
`-process_bulk_pins_for_body_bias {true | false}`	
	When set to `true`, rail analysis will process current sinks on the cell's bulk pin connections. This parameter is used during body bias analysis, where a body bias net is connected to bulk pins of the cell. The bulk pins for a cell is defined in the cell library using the `set_pg_library_mode -bulk_power_pins -bulk_ground_pins` command parameter. If bulk pins are defined using `set_pg_library_mode -power_pins -ground_pins`, then the `-process_bulk_pins_for_body_bias` parameter must not be used. *Default* : `false` *Data_type*: enum, optional
`-promote_pin_shapes_only {filler | decap | both}`	
	Specifies to promote either only filler shapes, only decap shapes, or both types of pin shapes during rail analysis. When this parameter is specified, the interface nodes for the filler and decap cells are ignored. The `-promote_pin_shapes_only` parameter helps to reduce the resistor and node count for big designs without affecting connectivity. This parameter must be specified with `set_rail_analysis_config -enable_2d_partition_extraction true` (enabled by default for the XP mode).
`-rdl_def` *def_file*	

	Specifies to instantiate the RDL instance/DEF at the top level and merge it with the DEF of the top level block being analyzed.
	The following is the usage model:
	`set_rail_analysis_config -rdl_def <def_file> -rdl_placement { X Y }`
	For GUI display, the same parameters also work for `read_def`:
	`read_def -rdl_def <def_file> -rdl_placement { X Y }`
	Data_type: string, optional

`-rdl_def_list {{rdl_filename1 x1 y1 orient1} {rdl_filename2 x2 y2 orient2} ...}`	
	Specifies to import the list of the RDL DEF files for the top-level rail analysis. This parameter allows to import multiple RDL DEFs with their location and rotation information in the top level. Example: `-rdl_def_list { {RDL1.def 0 0 N} {RDL2.def 100 200 N}}` *Data_type*: string, optional
`-rdl_placement {X Y}`	
	Specifies to place the RDL at a given X,Y location (in micron). *Data_type*: point, optional
`-rdl_orient { N|S|W|E|FN|FS|FE|FW }`	
	Specifies the orientation of the RDL DEF file. *Data_type*: enum, optional
`-read_thermal_map thermal_map_file`	
	Specifies to process the thermal map file to calculate resistance based on uniform temperature. Resistor scaling is supported for the tile-based power map file. Using the tile-base thermal map from Sigrity PowerDC, the software will scale the resistance value based on the temperature of the resistor's physical location, which provides more accurate IR and EM results with thermal impact. *Default* : `None` For more information on the thermal map file format, refer to the Thermal Map File section in the "File Formats" chapter of the *Voltus User Guide*. *Data_type*: string, optional
`-record_inst_peak_current {true | false}`	
	Specifies to save the instance peak current during rail analysis in the rail state directory. This parameter allows you to check and debug occurrences of abnormal instance peak current, which may result in high IR drop. The default value of this parameter is `false`. *Data_type*: enum, optional
`-record_results_start_time time`	
	Records rail analysis results based on the specified start time. The default unit of this parameter is `ns`. The software will use the specified time for all types of analysis, for example, instance voltage calculation, dynamic waveforms, and power-gate optimization. *Data_type*: float, optional
`-r_effective_detail_report {true|false}`	

	Specifies to include the following additional columns for instance pins in the effective resistance (`effr.rpt`) report: • `X` and `Y`: location (x/y coordinates) • `LAYER`: layer name • `PIN_NODES(FLOAT/TOTAL)` : number of floating nodes per total number of nodes of the instance By default, these columns are not included in `effr.rpt`. *Data_type*: enum, optional
`-r_effective_domain_report_format {1 | 2 | 3 | 4}`	
	Controls the format of the effective resistance analysis domain report (`domain_effr.rpt`). This parameter allows you to specify the level of detail to be displayed in the domain-based analysis report. The following table gives the report format for the four format values: **Format Value** / **Format** 1 (1 is the default format): `# PASS/FAIL REFF INSTANCE PIN(Pin REFF Percentage)` Example: `FAIL 10.0 instanceName VDD1(50.00%) VDD2(30%) VSS1( 10.00%) VSS2(10%)` 2: `# PASS/FAIL REFF INSTANCE POWER-PIN GROUND-PIN (Pin REFF Percentage)` Example: `FAIL 6.0 instanceName VDD1(83.33%) VSS1(16.67%)` 3: `# PASS/FAIL REFF INSTANCE PIN (Pin Absolute REFF Value, Pin REFF Percentage)` Example: `FAIL 10.0 instanceName VDD1(5.0, 50.00%) VDD2(3.0, 30%) VSS1( 1.0, 10.00%) VSS2(1.0 10%)` 4: `# PASS/FAIL REFF INSTANCE PWRNET PIN Layer X Y % GNDNET PIN Layer X Y %` Example: `FAIL 22.709 instanceName VDD pinName layerName 571.340 473.060 value(37.49%) VSS pinName Layername 571.340 471.800 value(62.51%)` *Data_type*: int, optional
`-r_effective_eval_nodes {port | tap}`	
	Default: `port` Reports effective resistance based on the port (pin) or tap nodes. By default, the software uses the port nodes for REFF reporting.
`-r_effective_pin_report_layer { top | bottom | all }`	
	Specifies which layer of the pin shapes are to be reported in the EFFR report. This parameter allows you specify which cell boundary pin layers will be used for the resistance report. The possible arguments of this parameter are: • `top`: Specifies to report only the pin shapes located on the highest pin layer. • `bottom`: Specifies to report only the pin shapes located on the lowest pin layer. • `all`: Specifies to report the pin shapes located on every layer. The default value is `all`. *Data_type*: enum, optional
`-r_effective_pin_report_method { best | worst }`	

	Specifies whether to report the best or worst effective resistance value of an instance pin with multiple nodes. The possible arguments of these parameters are: • `best`: Reports the smallest resistance value among the nodes of instance pin shapes for REFF. • `worst`: Reports the largest resistance value among the nodes of instance pin shapes for REFF. The default value is `worst`. *Data_type*: enum, optional
`-r_effective_report_all {true | false}`	
	Default: `false` Reports both fail and pass instance pins in the REFF report. By default, only the "FAIL" instance pins are reported.
`-remove_duplicate_inst {true | false}`	
	Specifies to remove duplicate cell instances during the DEF parsing stage of the rail analysis run. These duplicate cell instance names are from the same DEF file, and are also printed in a warning message so that you are not required to debug these disconnected instances. When this parameter is set to `true`, the duplicate cell instances within the same DEF file will not be reported as disconnected instances in the *Physically Disconnected Instances* section of the power-grid integrity report. The default value is `false`. *Data_type*: enum, optional
`-report_instances_missing_current_data {true | false}`	
	Specifies to generate a report that includes a list of all instances with missing current data. This parameter is supported in both the net-based and domain-based analyses. When this parameter is specified, the `<net_name>.nocurrentdata.inst.rpt` file is created in the `<state directory>/<net_name>/` directory. In addition, the log file includes the annotation statistics for instances with current data available on a per-net basis, as shown below: `#Instances missing current data: #count/#total_instances (%#percent value)` *Default* : `false` *Data_type*: enum, optional
`-report_layer_based_ir_average {true | false}`	
	Specifies to add the average IR drop value to the layer-based IR drop report and log file. *Data_type*: enum, optional
`-report_layers_above_pin_for_instance_ir {true | false}`	
	Specifies to report nodes above the pin layer. With this parameter set to `true`, the report will also contain layers above the pin nodes to the top most routing layer. The reporting will be based on the largest rectangular bounding box of the instance. *Data_type*: enum, optional
`-report_power_options` *option_names*	
	This parameter is used when `-report_power_in_parallel` is set to `true`. When the `-report_power_in_parallel` parameter is specified, you will not run the `report_power` command, therefore the reporting parameters cannot be specified. Using the `-report_power_options` parameter, you can specify the required `report_power` command parameters that are to be executed in parallel with rail analysis. *Data_type*: string, optional

`-report_power_domain {true | false}`	
	When set to `true`, the software generates an instance voltage file for each MSMV power domain in the design, e.g. VDD_VSS.iv and VDDm_VSS.iv where VDD and VDDm are MSMV power nets with common VSS net. *Default* : `true` *Data_type*: enum, optional
`-report_shorts {true | false}`	

	Specifies to check for power and ground shorts between two geometries belonging to different nets. The parameter performs power and ground short check for the following:
	• PG and PG nets
	• PG and signal nets
	• All extracted nets
	When the parameter is set to `true`, if shorts are detected, the file "`voltus_short.report`" will be generated in the extraction work directory. The format of the report file is:
	LayerName X-coord Y-coord NetName1 NetName2
	If one of the net is not an extracted net (that is, signal net or non-extracted PG net), it will be marked as N/A in the report.
	The following is the use model of the parameter:
	`set_rail_analysis_config -report_shorts true`
	The following is a snippet from `voltus_short.report`:
	`# Layer X Y NET NET`
	`AP 98.500 211.000 VSS N/A`
	`AP 98.500 219.550 VSS N/A`
	`AP 98.500 301.000 VDD25_LVTL N/A`
	`AP 98.500 309.550 VDD25_LVTL N/A`
	`M8 13007.800 2267.200 VSS VDD`
	`...`
	Data_type: enum, optional

`-report_power_in_parallel {true | false}`	
	Enables power analysis in parallel with rail analysis to improve the overall turnaround time. For information on the parallel flow setup, refer to the "**Run Dynamic Power Analysis and Rail Analysis in Parallel**" section in the *Dynamic Power and IRDrop Analysis chapter* of the *Voltus User Guide*. *Data_type*: enum, optional
`-report_voltage_drop {true | false}`	
	When set to `true`, instead of reporting the actual (effective) instance voltage, all the values will be reported in terms of voltage drop/bounce. For example, if the full pwr-gnd is 1V, pwr is 0.9 and gnd is 0.1, the IV file will report 0.8 (0.9-0.1) if this parameter is set to `false`, or 0.2 (0.1drop+ 0.1bounce) if the parameter is set to `true`. *Data_type*: enum, optional
`-reuse_state_directory` *dir_name*	

	Specifies an existing state directory. The software assumes that the design data (DEFs and GDS) and power-grid libraries are not changed, and will re-start analysis of the design using power-grid database inside the specified state directory. If the state directory does not contain the necessary power-grid database files, it will issue an error message and provide information on the missing files. For net based analysis, you should specify net-based state directory, as shown below: • `set_rail_analysis_config -reuse_state_directory VDD_25C_avg_1` For domain based analysis, you should specify the domain's state directory, as shown below: • `set_rail_analysis_config -reuse_state_directory domain_25C_dynamic_1` A new state directory will be generated for the new analysis with soft links to the old state directory files that were reused. *Data_type*: string, optional
`-rlrp_cell_report_layer` *filename*	

	Specifies the cell names and layer of the pin shapes that are to be reported in the RLRP report. This parameter allows you to perform cell-based layer reporting for instance RLRP report.
	The format of the specified input file is:
	`#cell_name #netname #pinlayer`
	`netname` is either a netname or "`ALL`". If you specify `ALL` for a net name, all the nets in the analysis domain will be considered for RLRP reporting.
	Example of an Input File
	`PDVDD_18_18_NT_DR_V ALL {M9,M1}`
	`PDVDD_18_18_NT_DR_V DVDD_D1 {M9,M1}`
	`PDVDD_18_18_NT_DR_V DVDD_D1 {M9,M1} VSS_D1 {M2}`
	Data_type: string, optional

`-rlrp_detail_report {true | false}`	
	Specifies to trace and report the total power gate resistance along the RLRP path for each instance. The RLRP report (`<netName>.rlrp_inst`) has the following additional columns for each instance: • `RLRP_TOTAL` - Total RLRP resistance including the power gate path (from the always-on and switched nets) • `NET_RES` - Total RLRP resistance excluding the power gate path • `PGATE_RES` - RLRP resistance for the power gate path • `PGATE_RON` - Number of power gate RON resistors The report allows you to trace the Ron resistors in each power gate instance and generate the RLRP report based on power gate instances. The default value of this parameter is `false`. *Data_type*: enum, optional
`-rlrp_eval_nodes {port | tap}`	
	Reports RLRP based on the port (pin) or tap nodes. By default, the software uses the port nodes for RLRP reporting. *Data_type*: enum, optional
-rlrp_percentage_threshold *value*	
	Specifies to output top x% of the instances with large RLRP. *Data_type*: int, optional
`-rlrp_pin_based_report` {true | false}	

	Specifies to generate a pin-based RLRP report. When specified, a new column "*Pin*" is included in the RLRP report. *Data_type*: enum, optional
`-rlrp_pin_report_layer { top | bottom | all }`	
	Specifies which layer of the pin shapes are to be reported in the RLRP report. This parameter allows you specify which cell boundary pin layers will be used for the resistance report. The possible arguments of this parameter are: • `top`: Specifies to report only the pin shapes located on the highest pin layer. • `bottom`: Specifies to report only the pin shapes located on the lowest pin layer. • `all`: Specifies to report the pin shapes with best or worst resistance value among all layers. The default value is `all`. *Data_type*: enum, optional
`-rlrp_pin_report_method { best | worst }`	
	Specifies whether to report the best or worst RLRP to an instance pin with multiple nodes. The possible arguments of these parameters are: • `best`: Reports the smallest resistance value among the nodes of instance pin shapes for RLRP. • `worst`: Reports the largest resistance value among the nodes of instance pin shapes for RLRP. • `eiv_worst`: Reports the RLRP of the node that has the worst EIV value among the tap nodes of instance pin shapes. • `eiv_best`: Reports the RLRP of the node that has the best EIV value among the tap nodes of instance pin shapes. The default value is `worst`. *Data_type*: enum, optional
-rlrp_threshold *value*	
	Specifies to ignore instances with RLRP less than the specified threshold value. *Data_type*: float, optional
`-rms_em_analysis {true | false}`	
	Specifies to calculate the RMS EM in the dynamic rail analysis run. When not specified, it will be enabled automatically if the EM model files are specified. You can set this parameter to `false` to skip the RMS EM analysis even if the EM models are provided. This parameter generates the `<net_name>.rj.rms.rpt` report. *Default* : `false` *Data_type*: enum, optional
`-write_voltage_waveforms {true | false}`	
	When set to `true`, instance voltage waveforms are saved in the state directory (.ptiavg). This waveform file can be loaded in the `view_dynamic_waveform` command to probe voltage waveform for desired instances. This waveform file can be fed to Tempus to perform SPICE critical path analysis with IRdrop waveforms. It can be used in Substrate Noise Analysis as well. *Data_type*: enum, optional
`-scale_initial_condition_current` *filename*	

	Specifies a file containing the current scale factor for the initial condition of the specified nets. The parameter allows you to increase/decrease the initial current at time 0 to minimize oscillation in the package and to shorten the settling time.
	The format of the file is:
	`default <default scale factor for all nets>`
	`<net name> <scale factor for this net>`
	`...`
	The use model of this parameter is:
	`set_rail_analysis_config -method dynamic -scale_initial_condition_current scale_factor.file`
	The content of *scale_factor.file* is:
	`default 1.0`
	`VDD 0.8`
	`VSS 0.7`
	Data_type: string, optional

`-seb_lifetime value`	
	Specifies the life time of product in hours. *Default* : `43800` hours (that is, 5 years) *Data_type*: int, optional
`-seb_table table_filename`	
	Specifies the file name of the Failures in Time (FIT) table. This file is provided by the foundry. The file has the following columns: • Layer • MTF • Sigma • Activate energy • Current density exponent *Data_type*: string, optional
`-seb_temperature temperature`	

	Specifies the temperature used to update the median time to failure (MTF) with respect to the ambient temperature.
	• If this temperature is specified, it would be used globally for all the resistors.
	• If this temperature is not specified, the seb_temperature may take the following possible values depending on the specific circumstances:
	  ◦ If detailed delta temperature file is not used for SEB, the `seb_temperature` always takes the same value as that used for EM analysis.
	  ◦ If detailed delta temperature file is used for SEB, the seb_temperature is calculated by the following equation:
	  `seb_temperature = env_temperature + delta_T`
	  Where delta_T is the maximum temperature rise, including joule heat and coupling heat from FEOL, in the resistor's bounding box with zero extension.
	Data_type: enum, optional

`-set_analyze_bbox {xmin ymin xmax ymax}`	
	Specifies the X, Y, min and max coordinates of the bounding box in which the instances can be analyzed. The values of the coordinates are specified in microns (um). *Data_type*: string, optional
`-shorting_resistance value`	

	Specifies the minimum shorting resistance value (in ohm) assigned to resistors. When specified, those resistors with resistance equal or less than the specified value are merged to reduce the size of the R network. *Default* : `0` *Data_type*: float, optional
`-skip_extraction {true | false}`	
	Specifies to reuse a work directory. It is recommended to use this parameter when you want to reuse the previous extraction work directory during the net-by-net rail analysis flow. *Default* : `false` *Data_type*: enum, optional
`-snap_layer_for_current_taps` *file_name*	

	Specifies the layer file name for adaptive current modeling. It snaps current taps from lower via layers to higher via layers for better performance. This feature is supported for EM view of both macro PGV and hierarchical PGV. The current taps from lower layers will be processed based on the specified tile size, and redistributed equally across all the vias within the tile at the higher via layers. The default tile size is `5umx5um (25 um2)` for macro PGV, and `25umx25um` (`625 um2`) for hierarchical PGV.
	For example, when running full chip analysis, you already have detailed analysis on the sub-blocks or hierarchical blocks so you might consider speeding up the analysis by merging the current taps from lower layer to spread it across higher via layers. The current drawn from within the PGV will still remain the same so the accuracy at the top level will not be affected.
	The format of the file is:
	#Instance Based
	`INST <inst_name> <via_layer_name> <tile_size>`
	#Cell Based
	`CELL <cell_name> <via_layer_name> <tile_size>`
	`via_layer_name` is the layer name used in the LEF file.
	`tile_size` is optional.
	Data_type: string, optional

`-static_multi_mode_analysis {true | false}`	

	Specifies to analyze the worst-case static IR drop across multiple functional modes of operation for macros. When this parameter is specified, Rail Analysis will use the multi-mode static current file (`power_static_multimode_scenario_file`) to iterate over multiple scenarios for all macro instances, and output the worst-case IR and EM report. The generated report identifies the worst Differential Instance Voltage (DIV) and the scenario that produces worst DIV across all instances.
	Default : `false`
	This parameter generates the following report in the Reports folder:
	`State_dir/Reports/powerNetName_gndNetName``.worst_scenario_div.rpt` - this file reports each instance's worst scenario. It lists the instances based on worst DIV in the ascending order to determine the worst among all worst instance DIVs.
	The following is a snippet from the worst scenario report:
	`#INST_NAME SCENARIO_ID DIV`
	`block19/ICG4/ICG 12 -0.107583`
	`block19/ICG3/ICG 12 -0.107557`
	`...`
	Here, `SCENARIO_ID` is the worst time in the current file.
	Data_type: string, optional

`-stream_file` *file*	Specifies a GDS file. Partial (RDL GDS in flip-chip) or full-chip GDS can be specified. You can view extracted parasitic network for GDS, but, in order to view GDS layout it needs to be imported as an OA database in Voltus. *Data_type*: string, optional

`-stream_map` *file*	Specifies the GDS layermap for the GDS file specified using `-stream_file` parameter. The GDS layermap file format is as follows:
	#layer_type layer_name gds *gds_layer_number <gds_data_type> <[port|text]>*
	`layer via8 gds 85`
	`layer metal9 gds 74`
	`layer metal9 gds 126 1 text`
	Data_type: string, optional

`-stream_offset` { *x y* }	
	Specifies the offset value used to adjust and shift the GDS of a Redistribution Layer (RDL) cell in the mixed DEF+GDS flow. This parameter is available only when the `-stream_file` parameter is specified. *Data_type*: point, optional
`-stream_purpose {metalFill | flipChip | fullChip}`	
	Specifies the data type of the GDS provided. • `metalFill`: The GDS data will be used as fill metals for manufacturing effects calculation but the physical shapes will not be extracted. • `flipChip`: The GDS data contains only the RDL layers, and the extracted physical shapes will be merged with the DEF netlist. • `fullChip`: The GDS data contains physical shapes for the entire design. *Data_type*: enum, optional
`-stream_top_cell` *cell_name*	
	Specifies the name of the GDS top cell. *Data_type*: string, optional
`-suppress_message` { *message_id* + }	
	The messages with the specified message ids will not be output. *Default* : No messages are suppressed. *Data_type*: string, optional
`-switching_bit_file_for_eiv_calculation` *filename*	

	Specifies the file that includes the cell or instance based switch-bit information. This parameter allows you to override the switch-bit in a PGV, or specify the switch-bit for a specific cell or instance.
	Note: The switch-bit information is defined for memory cells to distinguish between the pre-charge and functional mode (read/write) operation. During pre-charge, the memory cells may have large IRdrops due to large demand current in a short period of time. However, these cells can tolerate the large IRdrops because word-lines are shut-off. To accurately measure the effect of IRdrop on the operation of a memory, the instance voltage should be calculated during the functional mode. The switch-bit signal is used by the software to calculate the instance voltage when memory is in the functional mode.
	The format of the specified file is:
	`CELL | INST <cell_name | instance_name>`
	`<MODE_NAME> <Tmin1> <Tmax1> <Tmin2> <Tmax2>`
	where,
	• *cell_name*/*instance_name* must have a PGV with the dynamic current waveform.
	• `MODE_NAME` must match one of the mode names in the PGV for the cell.
	• `Tmin` and `Tmax` is the window when the switch-bit is ON, and when EIV will be calculated. The default time unit is ns.
	The Rail Analysis engine reads this switch-bit information, and reports the elapsed and worst instance voltage drop (IVD) when the switch-bit is 1. Rail Analysis will generate two IVD values for a given instance in the EIV report:
	• Worst IVD across simulation (`ELAPSE_EIV`)
	• Worst IVD in switching window (`WIN_EIV`)
	Data_type: string, optional

`-tmp_directory_name` *directory*	
	Specifies a directory to use for temporary files. You can also use the global environment variable `TMPDIR` to specify an alternative location for the temporary data. If both are specified, the directory specified using the `-tmp_directory_name` parameter will take precedence. Following is an example for setting the TMPDIR environment variable: `setenv TMPDIR ./temp_directory` *Data_type*: string, optional
`-temperature` *value*	
	Specifies the temperature in degree celcius during rail analysis. The default temperature is 25 degree celsius. *Data_type*: string, optional
`-top_cell_placement {X Y}`	

	Specifies to place the top DEF block at a given X,Y location (in micron) with respect to virtual top (0,0).
	The following is the usage model:
	`set_rail_analysis_config -top_cell_placement { X Y } -top_cell_orient { N|S|W|E|FN|FS|FE|FW }`
	For GUI display, the same parameters also work for `read_def`:
	`read_def -design_offset { X Y } -design_orient { N|S|W|E|FN|FS|FE|FW }`
	Data_type: string, optional

`-top_cell_orient { N|S|W|E|FN|FS|FE|FW }`	
	Specifies the orientation of the top DEF block. *Data_type*: string, optional

`-tsv_subckt_model_file_list` *filename*

Specifies the name of the external Through Silicon Via (TSV) subckt model files to be used for 3D-IC extraction. You can specify a list of files separated by space. Each file contains the subckt model information for TSV. The specified file has the Spice format.

A sample file is shown below.

```
.subckt TSV top bottom
R1 top n2 0.03075
R2 n2 bottom 0.03075
C1 n2 0 28f
.ends
```

`-unconnected_cell_ignore_file` *filename*

Specifies the name of a file containing the list of cells and instances to be ignored while reporting connectivity issues in the .rpt files, log file, and GUI. This parameter allows you to filter out the cells/instances that are known to have connectivity related issues and are not to be fixed.

The format of the file is:

```
CELL/INST    Cell Name/Instance Name    Pin Name
```

Here:

- `CELL` and `INST` are keywords
- Pin names are not mandatory. If pin names are not given, the corresponding cell/instance is ignored.

Example:

```
CELL  AND                => ignore all AND cell(s)
INST  INST1              => ignore instance INST1
CELL  BLOCK2*            => Ignore all blocks with the name starting with BLOCK2
CELL    *                => ignore all cells
INST    *                => ignore all instances
CELL   AN1  VDD*         => ignore AN1 cells's VDD* pins
INST  INST2 {VDD1 VDD2}  => ignore multiple pins of instance INST2
INST   INS*     VDD      => ignore VDD pin of all instances with the name starting with INS
```

`-unconnected_die_package_pins {ignore | error | edit}`

Specifies whether or not to ignore unconnected pins between die and package (if any), and consequently if the simulation should be paused, stopped, or continued. This option enables the manual connection between package and die through the *MCP Editor* utility if any disconnects are present.

The default value is `ignore`.

Data_type: enum, optional

`-use_early_view_list` *filename*

Specify cells in the file for which you want to use Early power-grid views. The file format accepts cell name in each line. You can specify wildcards (*, ?, -) in cell names.

Data_type: string, optional

`-use_em_view_list` *filename*

Specify cells in the file for which you want to use EM power-grid views. The file format accepts cell name in each line. You can specify wildcards (*, ?, -) in cell names.

Data_type: string, optional

`-use_ir_view_list` *filename*

Specify cells in the file for which you want to use IR power-grid views. The file format accepts cell name in each line. You can specify wildcards (*, ?, -) in cell names.

Data_type: string, optional

`-use_rms_delta_t {true | false}`	
	When this parameter is set to `true`, the `delta_T` used to calculate `seb_temperature` would be based on the RMS delta T (joule heat only). If this parameter is set to `false`, the `delta_T` used to calculate `seb_temperature` would be based on the sum of self-heat delta T and RMS delta T. *Default* : `false` *Data_type*: enum, optional
`-verbosity {true | false}`	
	Specifies to get side logs for extraction and rail analysis. The default value of this parameter is `false`. *Data_type*: enum, optional
`-verify_logical_connectivity {true | false}`	
	Specifies to report the list power and ground pins of each cell instance that do not have logical connectivity. The list of pins with missing logical connectivity are saved in a report file called `cell.unconnected_pins` under the extraction work directory. *Data_type*: enum, optional
`-voltage_source_search_distance` *value*	
	Specifies the search distance in microns for adding voltage sources at power pin locations. *Default* : `50` *Data_type*: float, optional
`-watch_location_waveform { {`*layerName xCoord yCoord*`}+ }`	
	Specifies to perform node based voltage probing for rail analysis. You can use this parameter to observe voltage waveform at a given location, specified by x,y coordinates and the metal layer. The unit of x,y coordinates is um, and the layer name should be a LEF/DEF layer. *Data_type*: string, optional
`-what_if_shapes_file` *filename*	
	Specifies to take the what-if shape specification from a file. This file is the file created with the `-save` parameter of the `create_what_if_shape` command. **Note**: If both the `-import_what_if_shapes` and `-what_if_shapes_file` parameters are specified, union of two sets of what-if shapes is considered for analysis. *Data_type*: string, optional
`-work_directory_name` *directory*	
	Specifies the work directory location for extraction data. When you specify this parameter, the software generates extraction temporary files in the specified directory. *Data_type*: string, optional
`-write_hier_block_boundary_voltage {true | false}`	

	Specifies to control the generation of block boundary voltage (BBV) for all hierarchical block instances that use xPGVs. The xPGVs for the blocks can be generated using the xPGV flow (Refer to "Hierarchical Power Integrity Analysis with Extreme Modeling" in the *Voltus User Guide*). The default value of this parameter is `false`. The BBV file contains realistic static or dynamic voltages for all nodes in the xPGVs starting where the top-level power-grid connects to the xPGVs. Since the BBV file is generated in context of full-chip analysis, running block-level analysis using this file gives insight to IR drop problems on the lower layers not included in the xPGV. This top-down flow can be used to identify and fix IR drop problems at the partition level without having to re-run the expensive fullchip analysis.
	
	`-write_hier_block_boundary_voltage` is a global parameter so all instances of xPGV will be generated. The output is written to the state directory for each power/ground net analyzed as '`instance_name.pp`'. The file contains '`putvsrc`' commands. You can output on a per-instance basis using the '`write block_boundary_voltage`' command in an include file, as shown below:
	`set_advanced_rail_options -voltus_rail_include_file_begin voltus_begin.inc`
	voltus_begin.inc:
	`write block_boundary_voltage` *`instance_name output_file_name`*
	For dynamic analysis, add `-watch`:
	`write block_boundary_voltage` *`instance_name output_file_name`* `-watch`
	The dynamic flow is recommended since transient information is not saved by default.
	
	To use the output BBV file at the block level, return to the block-level execution script and change `set_power_pads -format` to `boundary` and include the BBV file:
	`set_power_pads -net` *`net_name`* `-format boundary -file` *`bbv_file`*
	Example of a BBV file for dynamic analysis:
	* Boundary conditions for instance sample_instance
	`# putvsrc vsrc_# IR/Bounce x y PWLFILE pwl_file_for_node_location`
	`putvsrc K4 Vbc_VSS_1 1.210359e-01 247.296 179.712 PWLFILE`
	`sample_instance.pp.ptiavg 1`
	`putvsrc K3 Vbc_VSS_2 1.210360e-01 247.296 179.712 PWLFILE`
	`sample_instance.pp.ptiavg 2`
	`putvsrc K4 Vbc_VSS_3 1.210097e-01 247.296 175.104 PWLFILE`
	`sample_instance.pp.ptiavg 3`
	…
	
	Data_type: enum, optional

`-write_instance_reff_report` *`file`*	

	Generates a layer-based instance effective resistance report for specified cells/instances. The specified file contains the list of cells or instances for which the Reff report is to be generated. The format of the input file is:
	`CELL cell1`
	`CELL cell2`
	`INST inst1`
	
	The output file will be stored under each net's report folder `<net>.instance_effr.rpt`. For example, `Reports/VDD/VDD.instance_reff.rpt`
	
	For each cell or instance specified in the input file, the report has information such as layer-based effective resistance ranges, worst node location, average statistics, and interface node resistances. This feature is applicable for both net-based and domain-based analysis.
	
	Data_type: string, optional

`-write_instance_ir_report` *`file`*	

	Specifies to generate a layer-based IR drop report for a set of cells/instances. The *`file`* contains the list of cells or instances for which the IR drop report is to be generated. The format of the file:
	
	`CELL cell1`
	`CELL cell2`
	`INST inst1`
	
	The output file will be stored under each net's report folder `<net>.instance_ir.rpt`.
	
	Note: This feature is applicable for both net-based and domain-based analysis.
	
	Data_type: string, optional

`-write_instance_pin_ir_report` *file*	

	Specifies to generate a detailed report containing boundary node IR information for a set of cells/instances.
	The file contains the list of cells or instances for which the IR drop report is to be generated. The format of the file is:
	`INST <instance_name>`
	`CELL <cell_name>`
	The instance pin IR report will list all the pins' node information (layer, location) and the worst IR drop value.
	The output file will be stored under each net's report folder `NET/<net>.instance_pin_ir.rpt`. The following is a snippet of the IR drop report:
	`instance: <NAME>`
	`cell: <CELL>`
	`pin name: <PIN>`
	`# pin nodes: <total count>`
	`Min/Avg/Max: <min/avg/max interface node IR>`
	`For each of the interface node:`
	`IR X Y Layer`
	`......`
	The interface node output is sorted in the descending order of IR drop (that is, worst IR drop first).
	Data_type: string, optional

`-write_combinational_inst_voltage_drop_gif {true | false}`	
	Specifies to generate a combined instance voltage drop GIF file for all the nets specified in the domain. This GIF allows you to see the entire design's instance voltage plot. *Data_type*: enum, optional
`-write_demand_current_region_file` *filename*	

	Specifies a file that contains regions for custom demand current generation. This parameter allows to compute the total current within each given region, and sum up all tap currents that are outside the specified regions as a global port.
	The format of the region file is:
	`<REGION_PORT_LABEL> <XL YL XR YR> <NET>`
	here,
	• `REGION_PORT_LABEL` specifies a name for the region.
	• `XL YL XR YR` specifies the enclosing region coordinates in microns.
	• `NET` specifies the power/ground net name for analysis.
	Example file:
	`A 0 0 100 100 VDD_AO`
	When specified, one demand current waveform "`tran_<netname>_region.ptiavg`" will be generated per net in the rail output directory at:
	• For DP: `./Reports/Waveforms/tran_<netname>_region.ptiavg`
	• For XP: `./reports/rail/waveforms/tran_<netname>_region.ptiavg`
	Data_type: string, optional

`-write_package_pin_current_voltage {true | false}`	

	Specifies to generate a file containing transient current and voltage information of the package pins for the package analysis flow. This file is generated after dynamic rail analysis. The default value is `false`. This parameter is used in both the Sigrity Package Analysis (SPA) and non-SPA package analysis flows: • SPA flow - used in Sigrity tools to do chip-aware package analysis. This parameter is enabled by default (set to `true`) in the SPA flow. • Non-SPA flow - allows you to check the current on package pins. The format of the generated report in the static analysis mode (3 column format) is: *pin_name1 current1 voltage1* *pin_name2 current2 voltage2* ... The format of the generated report in the dynamic analysis mode is: .SUBCKT *Subcircuit_Name <list of pins for power and ground nets>* *MCP header with package pin name IA1 A1 0 PWL t1 i1 t2 i2 t3 i3 t4 i4 t5 i5 +t6 i6 … .ENDS *Subcircuit_Name* Here, • `IA1` (first item) is the current source element name • `A1` (second item) and `0` (third item) are the power and ground nodes • `PWL` keyword is followed by the time and current values *Data_type*: enum, optional
`-xpgv_config_file` *filename*	
	Specifies block details, such as, cell name for which xPGV is generated, via layer at which current is saved, block LEF file names, and block instance names for which xPGV is generated. This parameter is specified during block-level analysis. Configuration File Format: CELL <*block_name*> CUT_LAYER <*cut_via_layer_name*> Lef <*tech_lef_name block_lef_name*> END **Note**: This parameter is not supported with the `report_resistance` command. *Data_type*: string, optional
`-xp_cpu_per_job_power` *value*	
	Specifies the number of CPUs per power analysis job in the XP mode (`-enable_xp`). The specified value must be greater than `0`. This value should not exceed the value specified with `set_multi_cpu_usage -cpu_per_remote_host`. If this parameter is not specified, the value will be determined automatically based on the number of host resources. *Data_type*: string, optional
`-xp_cpu_per_job_simulation` *value*	
	Specifies the number of CPUs per simulation job in the XP mode (`-enable_xp`). The specified value must be greater than `0`. This value should not exceed the value specified with `set_multi_cpu_usage -cpu_per_remote_host`. If this parameter is not specified, the value will be determined automatically based on the number of host resources. *Data_type*: string, optional
`-xp_host_allocation_method {on_demand | at_startup}`	

	Specifies the method for remote host allocation during distributed processing. The possible arguments are: • `at_startup` – Specifies to wait for all available remote hosts before starting rail analysis. This parameter allocates all available remote hosts specified with `set_multi_cpu_usage` at startup. • `on_demand` - Allocates the available remote hosts based on demand. *Default* : `on_demand` *Data_type*: enum, optional
`-xp_purge {full}`	
	Specifies to clean up disk after run completion to delete temporary directories created for each analysis step (data generated in the voltus_data directory). If this parameter is specified, the reuse parameters (`-reuse_state_directory` and `-xp_reuse_extraction_directory`) will not work for the current state directory in the future runs. This parameter is applicable only in the XP mode (`-enable_xp`). *Data_type*: string, optional
`-xp_resume {true | false}`	
	Specifies to resume from the failed step, and skip the previously successful run steps. This parameter is applicable only in the XP mode (`-enable_xp`). *Default* : `false` *Data_type*: enum, optional
`-xp_reuse_extraction_directory` *directory_name*	
	Specifies the path to the state directory that contains the extraction power-grid database. This parameter allows you to skip extraction and reuse the extraction data from a previous run in the XP mode. *Data_type*: string, optional
`-xp_simulation_cpu_timeout` *value*	
	Specifies the time limit to either acquire the number of CPUs requested at the start of the run or acquire the number of CPUs specified with the `-xp_simulation_min_cpu` parameter in the XP mode. When this parameter is specified, the tool will wait for the given time limit to acquire the CPUs to run simulation before exiting. The unit for timeout is seconds. If a timeout value is not specified, the tool will wait for 7200 seconds (two hours) to acquire the CPUs. *Default* : `7200` *Data_type*: int, optional
`-xp_simulation_min_cpu` *value*	
	Specifies the minimum number of CPUs required for the rail simulation stage in the XP mode. The specified value must be greater than 0. When this parameter is specified, the tool waits to acquire the specified number of CPUs before performing simulation. This parameter can be used if you want to perform simulation with the minimum number of CPUs required to proceed with simulation instead of waiting for all the CPUs needed during the run. If this parameter is not specified, the tool performs simulation with the available number of CPUs. *Data_type*: int, optional

Examples

- The following command sets the rail analysis mode to static and xd accuracy, along with some other parameters:

```
set_rail_analysis_config \
-method static \
-accuracy xd \
-power_grid_library fast_library/fast_allcells.cl \
-voltage_source_search_distance 50
```

- The following command is an example of setting up rail analysis in the XP mode:

```
#resume previously failed XP run
set_rail_analysis_config -enable_xp true -xp_resume true

set_rail_analysis_config -method dynamic -accuracy hd \
...
-enable_xp true \
-xp_host_allocation_method at_startup \
-xp_reuse_extraction_directory dynamic_rail/core_125C_dynamic_1 \
-xp_cpu_per_job_power 2 \
-xp_cpu_per_job_simulation 8 \
-xp_purge full
```

- The following is an example of net-based early rail static analysis on an unplaced or partial placed design where virtual via/followpin would be created. The power would be distributed to placed instances on the basis of area. For unplaced instances, current region would be created on the M1 layer.

```
set_pg_nets -net vdd -voltage 0.998 -threshold 0.99301
set_power_pads -net vdd -format xy -file vdd.pp
set_power_data -format area -bias_voltage 0.998 -power 2.15
set_rail_analysis_config -method era_static -accuracy xd -extraction_tech_file ./qrcTechFile -era_current_distribution_layer M1 -
era_current_distribution unplaced -era_skip_virtual_via_by_type none -era_insert_virtual_followpin extended
report_rail -type net vdd
```

- The following is an example of domain-based early rail static analysis with virtual via/followpin. Here, user-specified instance current files are used.

```
set_pg_nets -net VSS -voltage 0 -threshold 0.18
set_pg_nets -net VDDm -voltage 1 -threshold 0.856
set_pg_nets -net VDD -voltage 0.84 -threshold 0.756
set_power_pads -net VDD -format xy -file vdd.pp
set_power_pads -net VDDm -format xy -file vddm.pp
set_power_pads -net VSS -format xy -file vss.pp
set_power_data -format current {instance_current_files/static_VSS.ptiavg instance_current_files/static_VDD.ptiavg
instance_current_files/static_VDDm.ptiavg  instance_current_files /static_VDDlu.ptiavg instance_current_files/static_VDDau.ptiavg}
set_rail_analysis_config -method era_static -accuracy xd -extraction_tech_file typical.tch -era_power_switch_file pwrgates_port.txt -
era_skip_virtual_via_by_type none -era_insert_virtual_followpin extended
set_rail_analysis_domain -name PD -pwrnets {VDD VDDm} -gndnets VSS
report_rail -type domain PD
```

- The following is an example of net-based early rail analysis with user specified ASCII instance and current region file. Here, all virtual vias and followpin creation have been skipped.

```
set_pg_nets -net vss -voltage 0.0 -threshold 0.01
set_power_pads -net vss -format xy -file vss.pp
set_power_data -format ascii -bias_voltage 1 -power 1.5 instance_power_file.txt
set_power_data -bias_voltage 1 -power 1.5
set_rail_analysis_config -method era_static -accuracy xd -extraction_tech_file ./qrcTechFile -era_current_region_file ./cur_region_file
-era_skip_virtual_via_on_layers MI M2
report_rail -type net vdd
```

- The following is an example of static early rail analysis with current distribution for unplaced instances by total die area:

```
set_rail_analysis_config -power_grid_library fast_library/fast_allcells.cl -extraction_tech_file typical.tch -accuracy hd -
ignore_shorts true  -method era_static -enable_rlrp_analysis true -voltage_source_search_distance 50  -era_current_distribution
unplaced -era_current_distribution_layer Metal1 -era_current_distribution_unplaced_area diearea -era_techlib_generation false -
era_check_wires_for_generated_current_regions true
```

- Let us consider a design with cell type A that has the M1 and M2 PG pin shapes, and cell type B that has the M3 and M4 PG pin shapes.
  The following command calculates the reff values of the interface nodes on the M2 pin shapes for the cell type A and on the M4 pin shapes for the cell type B, and reports the worst (biggest) value among them. Similarly, it calculates the rlrp values of the interface nodes on the M1 pin shapes for the cell type A and on the M3 pin shapes for the cell type B, and reports the best (smallest) value among them.

```
set_rail_analysis_config \
      -r_effective_pin_report_layer top \
      -r_effective_pin_report_method worst \
      -enable_rlrp_analysis true \
      -rlrp_pin_report_layer bottom \
      -rlrp_pin_report_method best \
…

report_resistance
```

- The following command reads the switch-bit information in the specified memory cells:

```
set_rail_analysis_config \
  -power_grid_library "${Libgen_Golden_Pgv}" \
  -accuracy hd \
  -ignore_shorts true \
  -method dynamic \
  -enable_rlrp_analysis true \
  -eiv_eval_window switching  \
  -limit_number_of_steps false \
  -watch_location_waveform u0 \
  -switching_bit_file_for_eiv_calculation  sw.list
```

  The following is a snippet of the sw.list file:

```
CELL MEM128Kx32
READ 0.1ns 0.13ns
WRITE 0.1ns 0.15ns
CELL MEM64Kx32
READWRITE  0.1ns 0.2ns 0.3ns 0.4ns
```

  The following is a snippet of the report:

```
INST_NAME WIN_EIV PWR_WIN_IV GND_WIN_IV WIN_TIME ELAPSE_EIV PWR_ELAPSE_IV GND_ELAPSE_IV ELAPSE_TIME CELL_NAME
*"NA" means the instance is disconnected from the net or the instance does not have timing/switching window
BEGIN
- external/MULT8_reg_3_6 0.75841 0.85220 0.09379 48.470 0.75841 0.85220 0.09379 48.470 SDFFQXL
```

- The following command specifies the instance-based PWL current file in the dynamic ERA flow:

```
set_rail_analysis_config -method era_dynamic -accuracy hd -vsrc_search_distance 500 -era_instance_dynamic_current_file
./inst_cell_pwl_file -power_grid_library ./../techonly.cl
```

  The following is a snippet of the PWL file:

```
instance ram/ram3 net VDD pwl (0ns 0mA 0.1ns 1mA 0.2nA 0.5mA) intrinsic_cap 10pf loading_cap 60pf
instance ram/ram2 net VDD pwl (0ns 0mA 0.1ns 1mA 0.2nA 0.5mA) intrinsic_cap 10pf loading_cap 60pf
...
```

- The following command specifies different temperature values to the layers M1, M2, M3, VIA1, VIA2, and VIA3 for average current limit analysis:

```
set_rail_analysis_config
...

   -em_temperature 110 \ #temperature for the remaining layers
   -em_temperature_layer_list {M1 125 M2 115 M3 105 VIA1 115 VIA2 105 VIA3 120} #layer-based temperature
```

  The log file will display the following info message:

```
Layer specific EM temperature will be used for AVG check for layer M1(125C), M2(115C), M3(105C), VIA1(115C), VIA2(105C), VIA3(120C).
```

  The header of the power EM report displays the layer-based temperature, as shown below:

```
# Temperature: 110.0 (C)
# Jmax_factor:
# ……
#   M2  0.981000 @ 115.0 (C)
#  ……
```

Related Information

- "Static Power, IRdrop and EM Analysis" in the *Voltus Stylus Common UI User Guide*
- "Dynamic Power and IRDrop Analysis" in the *Voltus Stylus Common UI User Guide*

set_rail_analysis_domain

```
set_rail_analysis_domain
-domain_name domain_name
-power_nets power_net_list
-ground_nets ground_net_list
[-threshold value]
```

Specifies a rail analysis power domain.

Note: For a power-gated domain, you should only specify the "always on" net. The switched net should not be specified as part of power domain definition, because it is traced by the program automatically.

Parameters

`-ground_nets ground_net_list`	
	Specifies a list of ground nets that are in the domain. *Data_type*: string, required
`-domain_name domain_name`	
	Specifies the name of the power domain. *Data_type*: string, required
`-power_nets power_net_list`	
	Specifies a list of power nets that are in the domain. *Data_type*: string, required
`-threshold value`	
	Specifies the minimum allowed voltage on the power net or the maximum rise on the ground net, depending on the setting of the `-voltage` parameter. The `-threshold`parameter is used to verify that the power grid passes the limit check and to set the filters for an IR plot. **Note:** This parameter does not support domains with MSMV power nets. If you have specified both net (`set_pg_nets -threshold`) and domain-based (`set_rail_analysis_domain -threshold`) thresholds, the net-based threshold will be honored. The valid value for threshold is between 0 and 0.5 (50%). For example, `set_rail_analysis_domain -threshold 0.1` indicates that 0.1 is 10% IRdrop. If a power net in the domain has nominal voltage of 1v, 10% of threshold will translate to 100mV of maximum effective IRdrop between the power and ground net. *Data_type*: float, optional

Examples

- The following command sets the power domain `TDSP` to include `VDD_TDSPCore_R` and `VSS` nets:
  ```
  set_rail_analysis_domain -domain_name TDSP -power_nets VDD_TDSPCore_R -ground_nets VSS
  ```

Related Information

- `set_rail_analysis_config`
- `set_pg_nets`
- `report_rail`
- "Static Power, IRdrop and EM Analysis" in the *Voltus Stylus Common UI User Guide*
- "Dynamic Power and IRDrop Analysis" in the *Voltus Stylus Common UI User Guide*

set_self_heat_analysis_mode

```
set_self_heat_analysis_mode
[-help]
[-alpha_parameters {{<layer_name> < α_overlapping > < α_connecting >}+}]
[-beta_parameters {C1 C2 C3}]
[-cell_thermal_resistance_file filename]
[-detail_delta_temperature_file filename]
[-detail_delta_temperature_region {x1 y1 x2 y2}]
[-die_instance_input_signal_em filename]
[-die_instance_name dieinstname]
[-instance_delta_temperature_file filename]
[-instance_power_file filename]
[-report_conn_pin_wire filename]
[-scale_rms_limit value]
[-tile_delta_temperature_file filename]
[-tiles {n_tile_in_x m_tile_in_y}]
```

Specifies the number of tiles, layer-based α coefficients, β coefficients, and list of cells with cell thermal resistances. This command will use the static power data to compute delta T FEOL for all instances in the design. This command:

- computes the RMS currents on signal wires
- computes the RMS currents on power network wires
- generates Detail Delta Temperature File (ddt), Instance Delta Temperature File (idt), and Tile Delta Temperature File (tdt)

The command supports both single-die and multi-die flows.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `set_self_heat_analysis_mode` parameter. For a detailed description of the command and all of its parameters, use the man command `man set_self_heat_analysis_mode`.
`-alpha_parameters {{<layer_name> < αoverlapping > < αconnecting >}+}`	
	Specifies the alpha parameters for all layers. It is the metal layer effect.
`-beta_parameters {C1 C2 C3}`	
	Specifies the thermal coupling effect coefficient Value.
`-cell_thermal_resistance_file filename`	
	Specifies the name of the cell's equivalent thermal resistance file. The specified file contains the list of cells and their corresponding thermal resistance value. The unit of thermal resistance is in °C/W (temperature rise per unit rate of heat dissipation). The format of the file is: `<cell_name> <thermal_resistance_in_C/W>` To get instance deltaT, use half of the instance power multiplied by the cell thermal resistance.
`-detail_delta_temperature_file filename`	
	Specifies the name of the generated BEOL detailed delta-temperature file.
`-detail_delta_temperature_region {x1 y1 x2 y2}`	
	Specifies the region coordinates for generating detailed delta temperature file.
`-die_instance_input_signal_em filename`	
	Specifies the setup Tcl script for each die instance.
`-die_instance_name dieinstname`	
	Specifies the die instance name for multi-die analysis. This option is only available in multi-die analysis (`set_rail_analysis_mode -die_mode multi-die`).

`-instance_delta_temperature_file` *filename*	
	Specifies the name of the generated FEOL instance-based delta-temperature file.
`-instance_power_file` *filename*	
	Specifies a file for instances and their corresponding static power. It is a two column format of <*instance_name*> and <*total_power*>.
`-report_conn_pin_wire` *filename*	
	Reports results on wires connecting to cell pins.
`-scale_rms_limit` *value*	
	Specifies the scale factor for the root-mean-square (rms) limit relaxation factor during Electromigration current limit analysis.
`-tile_delta_temperature_file` *filename*	
	Specifies the name of the generated FEOL/BEOL tile-based delta-temperature file.
`-tiles` *{n_tile_in_x m_tile_in_y}*	
	Specifies the number of tiles in the x and y directions in the tile-based delta temperature file.

Examples

- The following command is an example of `set_self_heat_analysis_mode` command:

```
set_self_heat_analysis_mode \
-die_instance_name die0 \
-alpha_parameters {{M1 0.74 0.85} {M2 0.62 0.85} {M3 0.53 0.85} {M4 0.42 0.85} {M5 0.32 0.85} {M6 0.26 0.85} {M7 0.26 0.85} {M8 0.26
0.85} {M9 0.26 0.85} {M10 0.26 0.85} {M11 0.26 0.85} {M12 0.26 0.85}} \
-beta_parameters {0.00423 -0.00045 0.81534}\
-instance_power_file ./dynamic_power.rpt \
-cell_thermal_resistance_file TRF.txt \
-instance_delta_temperature_file sh_itm.txt \
-tile_delta_temperature_file sh_ttm.txt \
-tiles {10 10} \
-detail_delta_temperature_file power_detail_die0.rpt
```

```
set_self_heat_analysis_mode \
-die_instance_name die1 \
-alpha_parameters {{M1 0.74 0.85} {M2 0.62 0.85} {M3 0.53 0.85} {M4 0.42 0.85} {M5 0.32 0.85} {M6 0.26 0.85} {M7 0.26 0.85} {M8 0.26
0.85} {M9 0.26 0.85} {M10 0.26 0.85} {M11 0.26 0.85} {M12 0.26 0.85}} \
-beta_parameters {0.00423 -0.00045 0.81534}\
-instance_power_file ./dynamic_power.rpt \
-cell_thermal_resistance_file TRF.txt \
-instance_delta_temperature_file sh_itm.txt \
-tile_delta_temperature_file sh_ttm.txt \
-tiles {10 10} \
-detail_delta_temperature_file power_detail_die1.rpt
```

```
set_self_heat_analysis_mode \
-die_instance_name die2 \
-alpha_parameters {{M1 0.74 0.85} {M2 0.62 0.85} {M3 0.53 0.85} {M4 0.42 0.85} {M5 0.32 0.85} {M6 0.26 0.85} {M7 0.26 0.85} {M8 0.26
0.85} {M9 0.26 0.85} {M10 0.26 0.85} {M11 0.26 0.85} {M12 0.26 0.85}} \
-beta_parameters {0.00423 -0.00045 0.81534}\
-instance_power_file ./dynamic_power.rpt \
-cell_thermal_resistance_file TRF.txt \
-instance_delta_temperature_file sh_itm.txt \
-tile_delta_temperature_file sh_ttm.txt \
-tiles {10 10} \
-detail_delta_temperature_file power_detail_die2.rpt
```

Related Topics

- "Self-Heating Effect Analysis" in the *Voltus User Guide*

set_voltage_regulator_module

```
set_voltage_regulator_module
-component_list {VRM_DEF_component_list}
-ground_pin_mapping {{spice_pin_name[DEF_net_name]}+}
-input_power_pin_mapping {{spice_pin_name[DEF_net_name]}+}
-name vrm_name
-netlist vrm_spice_netlist_file_name
-output_power_pin_mapping {{spice_pin_name[DEF_net_name]}+}
-reset
-sub_circuit_name vrm_subckt_name
```

Analyzes voltage regulator effects. Use this command to enable the Vreg flow.

You must specify this command before running the `analyze_rail` command.

Parameters

`-component_list {VRM_DEF_component_list}`
Specifies the names of the DEF instances which comprise the VRM. The component name must be the instance name used for the specified VRM. If this parameter is not specified, it is assumed that VRM is represented as a single MACRO on the die. *Data_type*: string, optional
`-ground_pin_mapping {{spice_pin_name[DEF_net_name]}+}`
Specifies the mapping between the ground pin name in the SPICE netlist and the corresponding DEF net name for the same power pin. *Data_type*: string, optional
`-input_power_pin_mapping {{spice_pin_name[DEF_net_name]}+}`
Specifies the mapping between the input power pin name in the SPICE netlist and the corresponding DEF net name for the same power pin. *Data_type*: string, optional
`-name vrm_name`
Specifies the VRM name assigned to a voltage regulator. You need to specify this parameter for each voltage regulator in the design. *Data_type*: string, optional
`-netlist vrm_spice_netlist_file_name`
Specifies the VRM netlist file in the SPICE format that is encapsulated in a single sub-circuit. This should also include SPICE models and testbench for each voltage regulator. *Data_type*: string, optional
`-output_power_pin_mapping {{spice_pin_name[DEF_net_name]}+}`
Specifies the mapping between the output power pin name in the SPICE netlist and the corresponding DEF net name for the same power pin. *Data_type*: string, optional
`-reset`
Resets all specified parameters back to default values. *Data_type*: bool, optional
`-sub_circuit_name vrm_subckt_name`
Specifies the subcircuit that corresponds to the VRM. This parameter is specified when the specified SPICE netlist has multiple subcircuit definitions. If this optional parameter is not specified, the first subcircuit found in the netlist file will be used. *Data_type*: string, optional

Examples

- The following command performs Vreg analysis for the VRM `vrm1`:

```
set_voltage_regulator_module -name vrm1 -netlist ./TEST.spi -input_power_pin_mapping {{VCC VCC}} -output_power_pin_mapping {{VDD VDD}}
-ground_pin_mapping {{ VSS VSS }} -sub_circuit_name LDO_MODEL -component_list { vmesh_reg_1 }
```

Related Information

- `set_rail_analysis_config`
- "Dynamic Power and IRDrop Analysis" in the *Voltus Stylus Common UI User Guide*

view_package_results

```
view_package_results
[-help]
[-result_file <filename>]
-type <package_analysis_type>
[-workspace <workspacename>]
```

Specifies to view the package IR drop analysis results in the PowerDC GUI which will be invoked from within Voltus/Innovus.

Parameters

`-help`	Outputs the command usage and a brief description about the command parameters.
`-result_file <filename>`	
	Specifies the name of the .xml result file. *Data_type*: string, optional
`-type <package_analysis_type>`	
	Specifies the type of package analysis. Currently, the software supports only the `dc` package analysis type. *Data_type*: string, required
`-workspace <workspace_name>`	
	Specifies the workspace name for package analysis (powerDC). The file name extension is `.pdcx`. *Data_type*: string, optional

Example

- The following example specifies to view the package analysis results:

  ```
  view_package_results -type dc
  ```

Related Information

- `set_advanced_package_options`
- `extract_package`
- "Package Analysis" in the *Voltus Stylus Common UI User Guide*

write_current_region

```
write_current_region
[-help]
-reset | {-region x1 y1 x2 y2
[-current value | -current_pwl_file filename]
[-die_instance_name dieinstname]
[-file filename]
[-layer layername]
[-intrinsic_capacitance value]
[-series_resistance value]
[-loading_capacitance value]
[-net net_name]
[-polygon {x1 y1 x2 y2 x3 y3 ..x1 y1}]
[-power_net net_name]
[-ground_net net_name]
}
```

Specifies static or dynamic current regions on global power-grids over the placed macro to accurately capture high and low current consuming regions during rail analysis.

You can use this command prior to rail analysis to specify current regions or power distribution over the top of power-grid on a user-defined layer. These current regions are generally specified for a placed macro for which power-grid view library is not available, and power/current distribution inside this macro could possibly impact IRdrop in neighboring regions. These current regions can also be defined over top-level power-grid for an unplaced macro during early stages of power-planning.

⚠ When you specify PWL currents over a region that includes instances, the current is distributed on nodes in the specified layer and not to each instance. The number of nodes to which current is distributed are determined by the tool so as to preserve the shape of the waveform.

Parameters

`-current value`	
	In the static mode (set_rail_analysis_mode -method static), the parameter specifies a single current value in mAmp. In the dynamic mode (set_rail_analysis_mode -method dynamic) the parameter specifies a PWL current waveform in time (in ns) and current (in mA) format. For example, 0ns 0mA 0.5ns 1mA 1ns 2mA 3ns 1mA 4ns 0. *Data_type*: string, optional
`-current_pwl_file filename`	

	Specifies a PWL current file. The file has two columns, time (in ns) and current value (in mA), as shown below:
	`time1 value1`
	`time2 value2`
	The `-current_pwl_file` and `-current` parameters are mutually exclusive.
	Data_type: string, optional

`-die_instance_name dieinstname`	
	Specifies the name of the die to which you need to apply a current region. This parameter is only applicable to the multi-die analysis flow (`set_rail_analysis_config -die_mode multi-die`). The parameter allows you to accurately assign current to power and ground nets in specific geometric regions for individual dies during rail analysis in 3D-IC designs.
`-file filename`	

	Specifies the name of a file containing the list of current regions for each die. `-file` can be used only with the `-die_instance_name` parameter. The other parameters of the `write_current_region` command can be included within the specified file.
	Example:
	`write_current_region -file write_current_region_die0.tcl -die_instance_name Die0`
	`write_current_region -file write_current_region_die1.tcl -die_instance_name Die1`
	`###File content in write_current_region_die0.tcl`
	`-net VDD -current 50mA -layer M2 -region "10 10 100 100"`
	`-net VDD -current 50mA -layer M2 -region "100 100 200 200"`
	`-net VSS -current 100mA -layer M1 -region "10 10 200 200"`
	`###File content in write_current_region_die1.tcl`
	`-net VDD -current 150mA -layer M2 -region "10 10 50 50"`
	`-net VDD -current 50mA -layer M2 -region "100 100 200 200"`
	`-net VSS -current 200mA -layer M1 -region "10 10 200 200"`

`-help`	Outputs a brief description that includes the type and default information for each write_current_region parameter. For a detailed description of the command and all of its parameters, use the man command man write_current_region.
`-ground_net net_name`	
	Specifies a ground net name on which the specified current is to be applied. *Data_type*: string, optional
`-intrinsic_capacitance value`	
	Specifies the cell intrinsic capacitance in pico farads (pf). This is required parameter when set in the dynamic mode (set_rail_analysis_mode -method dynamic). This parameter is disabled in static mode. *Data_type*: string, optional
`-layer layername`	
	Specifies a layer name on which the scaling will be performed. Multiple commands can be issued to scale resistance on multiple layers. This is a required parameter. *Data_type*: string, optional
`-loading_capacitance value`	
	Specifies the cell loading capacitance in pico farads (pf). This is required parameter when set in the dynamic mode (`set_rail_analysis_mode -method dynamic`). This parameter is disabled in static mode. *Data_type*: string, optional
`-net net_name`	
	Specifies a net name on which the specified current is to be applied. *Data_type*: string, optional
`-polygon {x1 y1 x2 y2 x3 y3 ..x1 y1}`	
	Specifies a rectilinear current region where the current will be distributed within. The rectilinear region enables you to specify multiple x,y points to add current regions in the areas that are not rectangular in shape. When specified, the last xy coordinate location will be the same as the first xy location. The xy coordinates will be in the counterclockwise sequence. *Data_type*: string, optional
`-power_net net_name`	
	Specifies a power net name on which the specified current is to be applied. *Data_type*: string, optional
`-region x1 y1 x2 y2`	
	Specifies the region on the layer across which the specified current and capacitance (in dynamic mode) is distributed by the rail analysis engine. The units of x and y should be in microns. *Data_type*: string, required

`-reset`	Resets all the previous write_current_region command settings. *Data_type*: bool, optional
`-series_resistance value`	
	Specifies the series resistance (ESR) value attached to each tap point in the current region definition. *Data_type*: string, optional

Examples

- The following command specifies static current region:
  ```
  write_current_region -current 2mA -layer METAL_1 -region "100 100 200 200"
  ```
- The following command specifies dynamic current region:
  ```
  write_current_region -current {0ns 0mA 0.5ns 1mA 1ns 2mA 3ns 1mA 4ns 0} -layer METAL_1 -region "200 200 400 400" -intrinsic_capacitance 40 -loading_capacitance 20
  ```
- The following command resets all write_current_region parameters to their default values:
  ```
  write_current_region -reset
  ```
- The following command specifies rectilinear current regions for die instance `Die1`:
  ```
  write_current_region -net VDD -current 50mA -layer Metal5 -polygon "10 10 100 10 100 100 10 100 10 10 " -die_instance_name Die1
  write_current_region -net VDD -current 50mA -layer Metal5 -polygon "100 100 200 100 200 200 100 200 100 100" -die_instance_name Die1
  ```

Related Information

- `scale_what_if_capacitance`
- `scale_what_if_resistance`
- `scale_what_if_current`
- "What-If Rail Analysis" in the *Voltus Stylus Common UI User Guide*

write_die_model

```
write_die_model
[-accuracy {xd | hd }]
[-output_dir directory_name]
-state_directory directory_name
[-rail_analysis_domain domain_name | -net net_name]
[-repeat_time time]
[-probing_node_file file_name]
[-use_sigrity_repeat]
[-max_ports_per_net integer]
[-enable_package_pin_map { true|false }]
[-enable_reference_node]
[-disable_current_source_node_zero {true | false}]
[-disable_global_node_zero { true|false }]
[-multi_die {true | false}]
[-port_grouping_by_ploc_files { true|false }]
[-port_grouping_file filename]
[-tmp_directory_name directory]
[-report_effective_rc]
[-probing_region_file filename]
[-write_effective_rc_deck]
[-port_grouping_tiles_file filename]
[-enable_xp {true | false}]
[-ignore_port_grouping_conflict]
[-generate_sipi_port_definition_file {true | false}]
[-grid_reduction_cut_layer layer_name]
```

Specifies to generate a reduced n-port die model in the form of equivalent die parasitic and current profile after rail analysis to perform system-level analysis with extracted package and board models. Resonance and electro-magnetic interference like system-level analyses can be performed using this die-model in package designing software like Allegro Package Design and Sigrity.

ⓘ This command requires a VTS-XL license.

Parameters

`-accuracy {xd | hd }`	
	Specifies the accuracy mode for die-model generation. *Default* : `hd` • `xd`: xd or accelerated definition accuracy mode is used for faster die model generation with slight reduction in RC accuracy. • `hd`: hd or high definition accuracy mode is used for most accurate die model generation. *Data_type*: enum, optional
`-disable_current_source_node_zero {true | false}`	

	Specifies to disable node zero for current sources in a generated die-model. The current source node zero is used to connect decoupling capacitance and the current source in the generated die-model. This parameter is added to generate a die model without current source node zero. By default, a virtual ground is used to connect decoupling caps and the current source in the generated die-model. When set to `true`, generates die model without the virtual ground and uses the ground network for connecting decoupling caps and current sources. It also creates the <BLOCK>_nportModel.sp.isrc_balanced file in the die-model directory. This file has information on the current sources and their port connections. The .subckt file generated for Spice simulation points to the .isrc_balanced file. *Default*: `false` *Data_type*: enum, optional
`-disable_global_node_zero { true|false }`	
	Specifies to disable the global node zero in a generated die model. • When set to `true`, generates die models without the virtual ground and uses the ground network for connecting decoupling caps and current sources. • When set to `false`, a virtual ground is used to connect decoupling caps and the current source in the generated die-model. *Default*: `true` *Data_type*: enum, optional
`-enable_reference_node`	
	Specifies to replace the global node zero with the reference port in the generated die model. *Data_type*: bool, optional
`-enable_package_pin_map { true|false }`	

	Specifies to support the 6-column style MCP format to map package pins to circuit ports. The following are the old and new MCP file formats:
	The original 5-column format is:
	<vsrc_name> <Ckt_port> <Net_name> <X coord> <Y coord>
	The new 6-column format is:
	<package_pin> <Ckt_port> <Net_name> <X coord> <Y coord> <vsrc_name>
	Example:
	Original format:
	`chip1/bmp_100_6_12 D1_bmp_100_6_12 VDDE 350.000 147.000`
	`chip2/bmp_122_3_15 D2_bmp_122_3_15 VDD2 321.000 983.800`
	New format:
	`bmp_100_6_12 D1_bmp_100_6_12 VDDE 350.000 147.000 chip1/bmp_100_6_12`
	`bmp_122_3_15 D2_bmp_122_3_15 VDD2 321.000 983.800 chip2/bmp_122_3_15`
	Data_type: enum, optional

`-enable_xp {true | false}`	
	Specifies to generate the die model in the Extensively Parallel (XP) mode in which processing is distributed over a large number of machines. *Default* : `false` *Data_type*: enum, optional
`-generate_sipi_port_definition_file {true | false}`	
	Generates a ports definition file during die-model creation. When set to `true`, the port definitions are output to the `port_defintion.txt` file, which is later used by Sigrity SystemPI for connecting the die model to the package design. *Default* : `false` *Data_type*: enum, optional
`-grid_reduction_cut_layer` *layer_name*	
	Specifies a cut via layer below which all the layers will be reduced. This feature minimizes data and enables efficient die model generation. *Data_type*: string, optional
`-ignore_port_grouping_conflict`	
	Specifies to not generate an error message in case there is a conflict between the user-defined port grouping and the default port groups (voltage sources that are grouped by mapping to the same package terminal). For example, vsrcA and vsrcB are both mapped to the package terminal PKG_T, but in the user-specified grouping file, vsrcA and vsrcB belong to two groups. When this parameter is specified, the tool will not generate an error message if there is a conflict, but instead will group the voltage sources according to the user-specified port grouping file. If this parameter is not specified and there is a conflict in port grouping, an error will be generated. *Data_type*: bool, optional
`-max_ports_per_net` *integer*	
	Specifies the maximum number of ports allowed for each net in a generated die model. If you do not specify this parameter, the software by default enables port grouping with number of ports set to `200`. If the number specified through this parameter is `0`, port grouping will be disabled. The maximum number of ports per net should be `0` or a positive value. A negative value will be treated as 0. **Note:** The `-max_ports_per_net` parameter is only meaningful for N-port die model creation. The 2-port die model creation always groups all ports together. *Data_type*: int, optional
`-multi_die {true | false}`	

	Specifies to generate a complete die model for a TSV/SiP design. *Default*: `false` *Data_type*: enum, optional
`-output_dir` *directory_name*	
	Specifies the directory in which you save the die-model files. *Data_type*: string, optional
`-port_grouping_by_ploc_files {true | false}`	
	Specifies to read the PLOC group information from the user-defined PLOC files. You can specify the port information in the PLOC file, as shown below: `Vsource_5_VDD 21985.00 23785.00 AP VDD p18` # with net and group `Vsource_7_VDD 22385.00 23785.00 AP VDD p18 C=1e-09 L=1e-11` # with capacitance and inductance When `-port_grouping_by_ploc_files` is set to `true`, the die model flow will read the user-specified port grouping information and generate a reduced die model. This parameter is mutually exclusive with the `-port_grouping_file` parameter. *Data_type*: enum, optional
`-port_grouping_file` *filename*	

	Specifies a text file containing the user-defined pin grouping details on a per power/ground net basis. When this parameter is specified, a reduced die model is generated that honors the user-specified port grouping. This die model has a reduced RC netlist and `.isrc` when compared with the original un-grouped die model. The degree of reduction will depend on how many pins are being grouped.
	The format of the pin grouping file is as follows:
	`NET <netname>`
	`<grouped pin name> <vsrc names which have to be grouped>`
	`<grouped pin name> <vsrc names which have to be grouped>`
	`NET <netname>`
	`<grouped pin name> <vsrc names to be grouped>`
	`<grouped pin name> <vsrc names to be grouped>`
	`END`
	where,
	• `NET` is a keyword followed by the power or ground net name
	• *`grouped pin name`* is the user defined port name
	• *`vsrc names`* are the actual design voltage sources that need to be grouped
	Data_type: string, optional

`-port_grouping_tiles_file` *filename*	

Specifies to group power ground ports by a user-defined tile array. All the ports in a specific tile are grouped together.

The use model of the parameter is:

```
create_die_model -port_grouping_tiles_file tiles.txt
```

The following is the format of the port grouping tiles format is:

```
NET <net>
BBOX <x1 y1 x2 y2>
ROWS <num rows>
COLUMNS <num columns>
```

Here,

- the keywords are highlighted in blue.
- *<net>* is the power supply net name
- *<x1 y1 x2 y2>* are the coordinates of the bounding box lower and upper corners. This is optional. If the bounding box is not specified, the default is the chip bounding box.
- *<num rows>* is the number of rows in the tile array
- *<num columns>* is the number of columns in the tile array

This parameter allows you to partition the voltage sources of the die model in to several port groups. If the same grouping pattern is used for the voltage sources of the die model and the package pins, it will help in matching the die ports and package terminals, and connecting the die model to the package netlist.

The grouping information is available in the MCP header of the generated die model file, as shown below:

```
* [REM]
* [Power Nets]
* vsrc_821 VDD2_GROUP_BUMP_tile01 VDD2 6316.25 20
* vsrc_820 VDD2_GROUP_BUMP_tile01 VDD2 623.75 20
* vsrc_822 VDD2_GROUP_BUMP_tile00 VDD2 5000 40
* vsrc_823 VDD2_GROUP_BUMP_tile00 VDD2 5002 40
……

*[Ground Net]
* vsrc_100 VSS_GROUP_BUMP_tile01 VSS 6300 25
* vsrc_101 VSS_GROUP_BUMP_tile01 VSS 6250 25
* vsrc_102 VSS_GROUP_BUMP_tile00 VSS 4000 40
* vsrc_103 VSS_GROUP_BUMP_tile00 VSS 4002 40
```

Here

- the first column is the voltage source name.
- the second column is the port name under which the voltage sources are grouped. It also displays the tile number to check a specific tile's power and group port during simulation.
- the third column is the net name.
- the fourth and fifth columns indicate the X and Y coordinates, respectively. The unit of x/y coordinate location is micrometer (µm).

Data_type: string, optional

`-probing_node_file` *file_name*

	Specifies the probe node file containing the probe point on a die. The file format for specifying probing nodes is as follows:
	`<user defined unique probe name> <location = x> <location = y> <layer> <Power/ground net name>`
	You can also specify the package node voltages and resistor currents that are to be probed in the following format:
	`<probe name> pkg ir <pkg circuit node name>`
	`<probe name> pkg rc <pkg circuit resistor name>`
	Example
	`probe_1 18 16 METAL_1 VDD`
	`# pkg node voltage`
	`probe_pkg_1 pkg ir GND.in0`
	`probe_pkg_6 pkg ir VDD.in2`
	`# pkg resistor current`
	`probe_pkg_cur_1 pkg rc R_1_1`
	`probe_pkg_cur_2 pkg rc R_5_2`
	Note: The units of x and y should be in microns.
	Data_type: string, optional

`-probing_region_file` *filename*	
	Specifies a probe node region file to define a probe region and use all points in the region as probe contact points on the die. The file format for specifying a probing region is as follows: *region_label x1 y1 x2 y2 layer_name net_name*
`-rail_analysis_domain` *domain_name* | `-net` *net_name*	
	Specifies the domain name or net name for which the die-model is to be generated. When the domain name is specified, coupled die-model is generated and it includes all power and ground nets associated with the specified domain. When net name is specified, decoupled die-model is generated for the specified net. Before specifying this command parameter, the power net name must be specified using the `set_pg_nets` command. This is a required option and one of them must be specified. *Data_type*: string, optional
`-repeat_time` *time*	Specifies the repeat time in ns. When this parameter is specified without the `-use_sigrity_repeat` parameter, the generated die model has the dynamic current profile of the die repeated to the specified time. By default, the die model consists of the default simulation period saved in dynamic current files during IR drop anlaysis. When this parameter is specified along with the `-use_sigrity_repeat` parameter, the software forces to write out repeating waveforms in the Sigrity specific format. If the `-use_sigrity_repeat` parameter is specified, the value specified through the `-repeat_time` parameter indicates the starting point of the repeating waveform. *Data_type*: float, optional

`-report_effective_rc`	Generates an additional report file (.txt) containing the effective resistance and capacitance information. The parameter is used for generation of a table for effective resistance and capacitance across multiple operating frequencies.
	The following is a sample report format:
	`# Net:VDD_AO, Net:VSS`
	`# frequency  # rdie  # cdie`
	`1.0000000e+07  4.73148443  2.47034747e-11`
	`1.2589254e+07  4.90209173  2.46916698e-11`
	`....`
	`....`
	`# Net:VDD_external, Net:VSS`
	`# frequency  # rdie  # cdie`
	`1.0000000e+07  -183.231085  6.65466673e-12`
	`1.2589254e+07  -164.007129  6.55112569e-12`
	`....`
	`....`
	Data_type: bool, optional

`-state_directory` *directory_name*	
	Specifies the top-level state directory that is generated by the static or dynamic IR drop analysis. This state directory contains analysis of all nets associated with the power domain; For example, `core_25C_dynamic_1`. *Data_type*: string, required
`-tmp_directory_name` *directory*	
	Specifies a directory to save temporary files for die-model generation. *Data_type*: string, optional
`-use_sigrity_repeat`	When this parameter is specified along with the `-repeat_time` parameter of the `write_die_model` command, the software forces to write out repeating waveforms in the Sigrity specific format. If the `-use_sigrity_repeat` parameter is specified, the value specified through the `-repeat_time` parameter indicates the starting point of the repeating waveform. If `-use_sigrity_repeat` is specified without specifying `-repeat_time`, time `0` is used for the starting point. The default unit for the value is "ns". **Note:** When the `-use_sigrity_repeat` parameter is specified, the software will write the stimulus current in the specific Sigrity format, which cannot be used by other Spice like simulators. *Data_type*: bool, optional
`-write_effective_rc_deck`	

	Specifies to include the Rdie/Cdie (effective resistance/capacitance) calculation for the die-model frequency domain simulation setup in the Spice (.sp) format.
	The following is a snippet of the .sp file with the frequency-domain setup for Rdie/Cdie calculation:
	`**************************************************`
	`* Frequency domain analysis to capture input impedance.`
	`.AC DEC 10 10Meg 5G`
	`.PRINT AC RDIE=PAR('VR(VSS_in,VDD_AO_in)')`
	`CDIE=PAR('1/(6.283185*freq*(VI(VDD_AO_in)-VI(VSS_in)))')`
	`**************************************************`
	Data_type: bool, optional

Examples

- The following command creates an `n-port` die model having the dynamic current profile of the die repeated upto `50ns`, using the state directory `allDomains_25C_dynamic_1` and saves the generated n-port die-model file in the `n_port_diemodel` output directory:

```
write_die_model
-state_directory irdrop_allOn/allDomains_25C_dynamic_1
-rail_analysis_domain allDomains -output_dir n_port_diemodel -repeat_time 50
```

- The following example shows the use model of the `-report_effective_rc` parameter to create a report file containing the effective resistance and capacitance information:

```
write_die_model
  -report_effective_rc -state_directory staticRailResults/ALL_125C_avg_1/
  -output_dir static_die_model -rail_analysis_domain ALL
```

Related Information

- `set_package`
- "Package Analysis" in the *Voltus Stylus Common UI User Guide*

write_hier_view

```
write_hier_view
-library_name name
[-output_dir dir]
[-block_lef lef_file | -block_stream gds_file]
[-block_lef lef_file | -stream_port_file gds_port_file]
[-libgen_command_file file]
[-max_via_cluster_mode {true | false}]
[-extraction_mode {fast | accurate}]
[-cell_name name]
[-def list_of_DEF_files]
[-oa_reference_libs OA_reference_libs]
[-power_grid_view_libraries primitive_cells.cl]
[-ground_pins {pin1 pin2 ...pinN}]
[-power_pins {pin1 value1 pin2 value2 ...pinN valueN}]
[-extraction_work_directory path_to_extraction_work_directory]
[-compress_power_grid_db {true | false}]
[-parasitic_extractor_command_file filename]
[-cluster_via1_ports {true | false}]
[-cluster_via_rule { {via_layer1 number_of_equidistant_vias}...}]
[-cluster_via_size value]
[-ignore_fillers {true | false}]
[-ignore_decaps {true | false}]
[-ignore_shorts {true | false}]
[-force_library_merge {true | false}]
```

Creates a hierarchical power-grid view. The hierarchical PGV generation of a block is part of the top-level Hierarchical Rail Analysis.

Parameters

`-block_lef lef_file`	
	Specifies the name of the block LEF file. The `lef_file` must have all power pin port geometries defined for the hierarchical block. *Data_type*: string, optional
`-block_stream gds_file`	
	In absence of block LEF, you can specify block GDS and port label file to generate hierarchical power-grid view. *Data_type*: string, optional
`-cell_name name`	Specifies the block cell name as specified in the block DEF file. *Data_type*: string, optional
`-cluster_via1_ports {true | false}`	
	Designs with follow-pin routing on M1 and M2 layers have standard cells library LEF with M1, M2, and VIA1 ports. This parameter specifies to cluster VIA1 ports for such cells in order to improve overall extraction and rail analysis performance. *Default*: `true` This parameter is set to `true` by default to improve VIA1 port handling in designs with M1 and M2 follow-pin routing. This parameter gives significantly better performance by clustering equidistant VIA1 ports defined in cell LEF without significant loss of accuracy. However, if you specify a clustering rule, the user specification takes precedence over automated VIA1 clustering. *Data_type*: enum, optional

`-cluster_via_rule { {`*`via_layer1 number_of_equidistant_vias`*`}...}`	
	Controls the number of vias to cluster on a layer basis. The VIA clustering rule specified using this parameter will override the default clustering rule for a given accuracy mode. The following command will cluster 100 equidistant VIA1 cuts, 200 equidistant VIA2 cuts, and 300 equidistant VIA7 cuts: `write_hier_view -cluster_via_rule { {VIA1 100}} {VIA2 200} {VIA7 300}}` The rest of the VIAs will be clustered using the default clustering rule depending upon the rail analysis accuracy mode. *Data_type*: string, optional
`-cluster_via_size` *value*	
	Specifies the size of the via array to cluster all via layers except VIA1 (layer between Metal1 and Metal2). The default value is `625` (25x25 via array), except when running static analysis using HD mode. The default for static HD mode is `16` (4x4), primarily to preserve more accuracy for EM analysis. *Default*: `625` *Data_type*: int, optional
`-compress_power_grid_db {true | false}`	
	Specifies to compress the output files generated by extraction and rail analysis to save disk space usage. This parameter reduces disk space requirement from the extraction and rail analysis stages at approximately 5-10% performance penalty. *Default*: `false` *Data_type*: enum, optional
`-def` *list_of_DEF_files*	
	Specifies a list of block DEF files for hierarchical analysis. *Data_type*: string, required
`-extraction_mode {fast | accurate}`	
	Determines what power-grid views will be created, dependent on whether fast or accurate extraction is to be done. By default, hierarchical PGV generates both detailed and reduced views. *Data_type*: enum, optional
`-extraction_work_directory` *path_to_extraction_work_directory*	
	Specifies to re-use the extracted power-grid database to generate HPGV. This parameter is only supported to generate hierarchical extraction view of a block in the design. The specified extraction work directory must contain the extracted data for all the nets for which the hierarchical extraction view needs to be generated. This parameter is mutually exclusive with the `-def` parameter of the `write_hier_view` command. *Data_type*: string, required
`-force_library_merge {true | false}`	

	Specifies to force merge PGVs with different resistivity and layers. It uses the technology information of the first library. If multiple tech-only PGVs are specified, it will take the first tech-only PGV definition and the first cell definition. If you are merging a tech-only PGV with other PGVs, it will keep the tech-only technology definition and the first cell definition. *Data_type*: enum, optional
`-ground_pins {pin1pin2...pinN}`	
	Specifies a list of ground nets of the block. *Data_type*: string, optional
`-ignore_decaps {true | false}`	
	Ignores decap cells during rail analysis. It ignores cells that are tagged as DECAP cells during library characterization or set as decap cells using the `set_rail_analysis_mode -decap_cell_list` command parameter. If decaps are used to preserve connectivity on the follow-pin routing, you must set this parameter to `false`. Ignoring decaps during dynamic analysis is not recommended. *Default*: `true` for static, `false` for dynamic *Data_type*: enum, optional
`-ignore_fillers {true | false}`	
	Ignores filler cells during rail analysis if the list of filler cells are defined in PGV generation, or specified as fillers using the `set_rail_analysis_mode -filler_cell_list` command parameter. During rail analysis, ignore library cells that are tagged as FILLER cells and have no capacitance associated with the interface nodes. If fillers are used to preserve connectivity on the follow-pin routing, you must set this parameter to `false`. *Default*: `true` *Data_type*: enum, optional
`-ignore_shorts {true | false}`	
	Ignores shorts that are found during extraction and continues hierarchical PGV generation. However, `-ignore_shorts` cannot be applied if the design contains GDS. *Default*: `false` *Data_type*: enum, optional
`-libgen_command_file file`	
	Specifies the file name of a power-grid library generation (libgen) command file. When specified, it will be included in the auto-generated `libgen.cmd` file at the end as `include <file>`. This parameter allows you to provide a file that includes specific library generation options. *Data_type*: string, optional
`-library_name name`	
	Specifies the name of the library that the block power-grid view will be placed in. A `.cl` extension will automatically be added to the name. *Data_type*: string, required
`-max_via_cluster_mode {true | false}`	

	Specifies the extractor to cluster vias automatically to reduce the RC elements by clustering vias into one node, therefore reducing the IR drop analysis runtime in dynamic mode. It is based on the notion that current evenly distributes through all vias of an array via. *Default*: `true` *Data_type*: enum, optional
`-oa_reference_libs` *OA_Reference_libs*	
	Reads OpenAccess reference libraries. *Data_type*: string, optional
`-output_dir` *dir*	Specifies the output directory that the results will be saved in. *Data_type*: string, optional
`-parasitic_extractor_command_file` *filename*	
	Specifies the name of the file input to the extractor invoked by the Library Generation engine. This file contains information about the cells in the design and is used for resistance and capacitance extraction. For variables and functions that can be placed in this command file, see *Extractor Options* in *Voltus Text Command Reference*. In general, this parameter is used for debugging purposes. *Data_type*: string, optional
`-power_grid_view_libraries` *primitive_cells.cl*	
	Specifies a list of primitive cell libraries for hierarchical PGV generation. *Data_type*: string, optional
`-power_pins {`*pin1value1pin2value2 ...pinNvalueN*`}`	
	Specifies a list of power nets and the nominal voltage of those nets. *Data_type*: string, optional
`-stream_port_file` *gds_port_file*	
	In absence of block LEF, you can specify block GDS and port label file to generate hierarchical power-grid view. *Data_type*: string, optional

Examples

- The following command generates a hierarchical power-grid view for the hierarchical block using LEF and extraction directories for VDD and VSS nets:

  ```
  write_hier_view -block_lef blockA.lef -extraction_work_directory work -library_name blockA  -power_grid_view_libraries
  ./pgv/stdcells.cl  -power_nets {vdd 1.2} -ground_nets {gnd} -cell_name blockA
  ```

  The design.lef must have all power pin port geometries defined for the hierarchical block. The generated hierarchical PGV library is blockA.cl. The library is in binary format and it stores the block's connectivity to the top-level based on the provided LEF data. It stores the extracted power-grid network and current distribution from the provided directories.

- The following command specifies to force merge PGVs with different resistivity and layers:

  ```
  write_hier_view  \
    -force_library_merge true \
    -library_name lib1  \
    -view hier_extraction_view \
    -power_grid_view_libraries {../accurate_stdcells_A.cl ../accurate_stdcells_A.cl ... ../accurate_stdcells_AN.cl } \
    -def {../design.def.gz} \
    -cell_name  CELLA \
    -power_pins {VDD 1.32} \
    -ground_pins {VSS} \
    -block_lef {./design.lef}  \
    -extraction_mode accurate \
  write_pg_library -output pgv
  ```

Related Information

- set_rail_analysis_config
- set_dynamic_rail_simulation
- report_rail
- "Hierarchical Rail Analysis" in the *Voltus Stylus Common UI User Guide*

write_power_pads

```
write_power_pads
[-append_to_voltage_source_file]
[-auto_fetch]
[-cell_list {cell_list} ]
[-cell_pin {{cellname1 pinname1}...{cellnamen pinnamen}+} ]
[-clear]
[-display]
[-format {padcell xy tsv}]
[-honor_pin_connection]
[-instance_list {inst_list} ]
[-instance_pin {{instname1 pinname1}...{instnamen pinnamen}+} ]
[-layer {layername}]
[-net netname]
[-package_resistance value ]
[-package_capacitance value ]
[-package_inductance value ]
[-region {x1 y1 x2 y2}]
[-region_pitch {xpitch ypitch}]
[-snap_distance { true|false }]
[-voltage_source_file vsrc_filename]
[[-add | -delete] -location {x y}]
[-list]
```

Returns a list of DC sources from your design. DC sources are identified in the LEF file as CLASS PAD POWER, or in the DEF file as PIN with +USE POWER/GROUND. If a single port shape is connected to N number of wires, N number of DC sources are reported for that port shape.

You can use one of the following three methods to fetch voltage sources into the database:

- fetch all voltage sources
- fetch the voltage sources within the specified region of the specified layer controlled by the specified xPitch/yPitch
  If the voltage sources with the specified xPitch/yPitch are not on the stripe, you can use the `-snap_distance` parameter to snap the voltage sources to the stripe. As a result, the voltage sources are generated only on the stripes and the xPitch/yPitch is honored. Use this feature to generate mesh type voltage sources on any process layer.
- fetch voltage source for the specified cell/cell pin/instance/instance pin

You can edit the fetched voltage sources before saving them to a voltage source file. To save the fetched list to a file, use the `-voltage_source_file` parameter. By default, voltage sources will not be displayed on GUI after they are saved to file.

ⓘ The `write_power_pads` command supports hierarchical IR drop analysis for a TSV design. When you specify the `write_power_pads` command to fetch the voltage sources from the LEF file, the command not only searches for the P/G pin of the pad connected to the specified P/G net, but also fetches for the embedded bump TSV property definition for the P/G pin connected to the P/G net. The port that contains the point defined in the embedded bump property is considered as the embedded bump.

Parameters

`-add`	Adds the location of a new voltage source to memory. *Data_type*: bool, optional
`-append_to_voltage_source_file`	Appends the fetched voltage sources to the existing voltage source file. You can use this parameter to fetch voltage sources in the incremental mode. *Data_type*: bool, optional
`-auto_fetch`	Specifies to fetch all voltage sources. *Data_type*: bool, optional

`-cell_list {cell_list}`	
	Specifies the cells from which voltage sources are determined. *Data_type*: string, optional
`-cell_pin {{cellname1pinname1}...{cellnamenpinnamen}+}`	
	Specifies the cells and their pins from which voltage sources are determined. *Data_type*: string, optional
`-clear`	Clears the display of the DC sources that were specified in the *Edit Pad Location* form. *Data_type*: bool, optional
`-delete`	Deletes the location of a voltage source from memory. *Data_type*: bool, optional
`-display`	Loads and displays the pad location data from the specified file. *Data_type*: bool, optional
`-format {padcell | xy | tsv}`	
	The format of the saved pad location file. The choices are Pad File (VoltageStorm), XY File (xy coordinates), and TSV. You can create and save the voltage sources location file with a unique name for the voltage sources (both VDD and VSS). *Default*: `xy` *Data_type*: enum, optional
`-help`	Outputs the command usage and a brief description about the command parameters.
`-honor_pin_connection`	
	Honors pin wire connection when fetching voltage source location from a power pad. When you specify this parameter, only pins with wire connection are output as voltage sources. You have the choice to specify whether real wire connection is to be honored or not. Typically, during the sign-off stage, power pads are already fully routed so the unconnected power pads should not be counted in as voltage sources. You can specify this using the -honor_pin_connection parameter. Alternatively, in the early stage, power pads may not have been fully routed so you do not need to set this parameter. **Note:** For a port with shapes at multiple layers, all layer connected shapes are fetched as voltage sources. *Data_type*: bool, optional
`-instance_list {inst_list}`	
	Specifies the instances from which voltage sources are determined. *Data_type*: string, optional
`-instance_pin {{instname1pinname1}...{instnamenpinnamen}+}`	
	Specifies the instances and their pins from which voltage sources are determined. *Data_type*: string, optional
`-layer` *layerName*	
	Specifies the process layer in LEF on which the voltage sources are to be generated. When the `-layer` parameter is specified in conjunction with the `-region {x1 y1 x2 y2}` and `-region_pitch {xpitch ypitch}` parameters, the layer set here is the layer for which voltage sources are to be fetched. If the `-layer` parameter is not used together with the `-region {x1 y1 x2 y2}` and `-region_pitch {xpitch ypitch}` parameters, the layer set here is for the lowest layer to snap (the default value is `M5`). *Data_type*: string, optional
`-list`	Loads and displays all voltage sources from memory. *Data_type*: bool, optional

`-location {x y}`	Specifies the location at which the specified voltage source is to be added or deleted. *Data_type*: point, optional
`-net netname`	Specifies the name of a power net for which to list DC sources. *Default:* If you do not specify a net name, the power analyzer automatically lists DC sources for all nets. *Data_type*: string, optional
`-package_resistance value`	
	Specifies the resistance value of the package resistance-inductance-capacitance (RLC) model in Ohm. *Data_type*: float, optional
`-package_capacitance value`	
	Specifies the capacitance value of the package resistance-inductance-capacitance (RLC) model in Farad (F). *Data_type*: float, optional
`-package_inductance value`	
	Specifies the inductance value of the package resistance-inductance-capacitance (RLC) model in Henry (H). *Data_type*: float, optional
`-region {x1 y1 x2 y2}`	
	Generates the voltage sources in the specified region. *Data_type*: rect, optional
`-region_pitch {xpitch ypitch}`	
	Specifies the region pitch in the x and y direction for the voltage sources. *Data_type*: point, optional
`-snap_distance value`	
	Specifies the snapping distance constraint from IO/pad pins. If this parameter is not specified with a value, no snapping will be done. *Default:* 0 Specifies whether to snap DC sources from I/O pad pins to some location on the power structure. When set to true, the software performs autofetch with snapping. For I/O pad pins that are connected to core ring, the software selects the highest layer and center of the pin as the DC source location. When I/O pad pins are not connected to core ring, the software snaps the location of the I/O pad to somewhere on the core ring. The DC source location should be within the distance of snapping distance constraint from I/O pad pins, and the layer should not be lower than the lowest layer snap constraint. *Data_type*: float, optional
`-voltage_source_file vsrc_filename`	
	Specifies the name of the voltage source file that is required to save the auto fetched result. *Data_type*: string, optional

Examples

- The following command fetches all voltage sources of the net `VDD`:
  ```
  write_power_pads -net VDD -auto_fetch
  ```
- The following command loads and displays all voltage sources from memory:
  ```
  write_power_pads -list
  ```
- The following command specifies the location at which the specified voltage source is to be added:
  ```
  write_power_pads -add -net VDD -layer METTOP -location {500 500}
  ```
- The following command specifies the location from which the specified voltage source is to be deleted:
  ```
  write_power_pads -delete -location {500 500} -layer METTOP -net VDD
  ```
- The following command fetches the voltage sources within the specified region of the layer `METAL6` controlled by the specified xpitch/ypitch, and saves the sources to the file `report2` in the xy format:
  ```
  write_power_pads -net VDD -region 100 100 3000 3000 -region_pitch 555 444 -layer METAL6 -format xy -snap_distance 100 -
  voltage_source_file report2.pp
  ```

  ⚠ In the first example, the output was not saved but it remains in the memory till you specify the second example. Before loading and saving the output of the second example, the first example output in the memory is saved to an auto generated dc source file incrementally, like pp.src, pp1.src, pp2.src, and so on.

- The following command fetches voltage source for the cell BUMPCELL in the xy format:
  ```
  write_power_pads -net VDD -cell_list BUMPCELL -format xy
  ```
- The following command clears the voltage sources displayed in the GUI and the fetched voltage sources in the database:
  ```
  write_power_pads -clear
  ```
- The following command displays the fetched and saved voltage sources for the net `VDD` in the file `report2` :
  ```
  write_power_pads -net VDD -auto_fetch -voltage_source_file report2.pp -display
  ```
- The following command displays voltage sources in the GUI by loading the saved voltage source file `report1`:
  ```
  write_power_pads -voltage_source_file report1.pp -display
  ```
- The following command saves the fetched voltage sources to the voltage source file `report1.pp`:
  ```
  write_power_pads -voltage_source_file report1.pp
  ```

Related Information

- `set_rail_analysis_config`
- `report_rail`
- "Static Power, IRdrop and EM Analysis" in the *Voltus Stylus Common UI User Guide*
- "Dynamic Power and IRDrop Analysis" in the *Voltus Stylus Common UI User Guide*

write_power_up_report

```
write_power_up_report
[-help]
[-switch_net net_name]
-output_directory directory_name
-state_directory directory_name
```

Specifies to generate power-up statistics report to view the ramp up/inrush current, worst power switch voltage, and additional instance voltage data (average, median, max, and standard deviation voltage data versus time) for the powering-up rail. The following files are generated in the GIF and text formats:

- average voltage of instances (.avgV)
- max voltage of instances (.maxV)
- min voltage of instances (.minV)
- median of instance voltages (.medianV)
- standard deviation of instance voltages (.deltaV)
- total inrush current (.inrushI)
- average voltage of power gates (.avgV)
- max voltage of power gates (.maxV)
- min voltage of power gates (.minV)
- median of power gate voltages (.medianV)
- standard deviation of power gate voltages (.deltaV)

Parameters

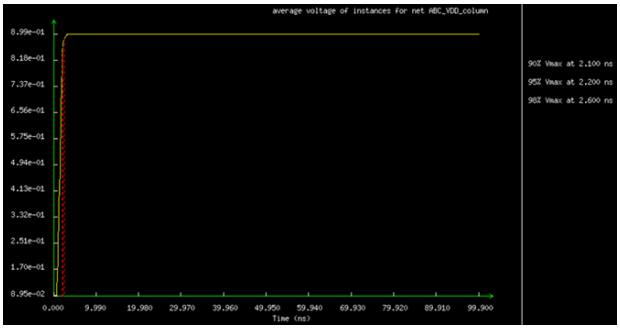
`-switch_net net_name`	
	Specifies the name of the powering up rail for which you want to generate detailed power-up statistics. *Data_type*: string, required
`-output_directory directory_name`	
	Specifies the output directory in which the power-up statistics results will be saved in. *Data_type*: string, required
`-state_directory directory_name`	
	Specifies the top-level state directory that is generated by IR drop analysis. This state directory contains analysis of all nets associated with the power domain; for example, core_25C_dynamic_1. *Data_type*: string, required

Examples

- The following command generates power-up statistics for the `ABC_VDD_column` net:
  ```
  write_power_up_report -switch_net ABC_VDD_column -state_directory core_25C_dynamic_1 -output_directory rpt
  ```

  The following is a snippet of the text and GIF formats with information on average voltage of instances:

Text Format (`ABC_VDD_column.avgV`)	**GIF Format (`ABC_VDD_column.avgV`)**

`#List average voltage of instances for net ABC_VDD_column`
`# 90% of nominal voltage at 2.1 ns`
`# 95% of nominal voltage at 2.2 ns`
`# 98% of nominal voltage at 2.6 ns`
 	
`#XLABEL# Time (ns)`
`#YLABEL# Voltage (V)`
`0 0.0894674`
`0.1 0.0894675`
`0.2 0.0894676`
`...`
	

Related Information

- `set_rail_analysis_config`
- "Power Gate Analysis" in the *Voltus Stylus Common UI User Guide*
- "Dynamic Power and IRDrop Analysis" in the *Voltus Stylus Common UI User Guide*

32

RC Extraction Commands and Attributes

- create_parasitic_coordinate_transform
- extract_rc
- extract_rc Category Attributes
- read_parasitics
- read_rc
- read_spef
- report_net_parasitics
- report_rcdb
- report_rc_factors
- report_unit_parasitics
- reset_parasitics
- set_extract_shrink_factor
- write_cap_table
- write_extraction_spec
- write_parasitics
- write_rcdb

create_parasitic_coordinate_transform

```
create_parasitic_coordinate_transform
[-help]
{-inst inst_name
[-x_block_origin value]
[-x_offset value]
[-y_block_origin value]
[-y_offset value]
[-orient {R0 R90 R180 R270 MY MX90 MX MY90}]
[-relative_to_parent]} | {-reset}
```

Determines accurate location information of nodes during SPEF reading.

In case of hierarchical designs, there can be multiple blocks in a design that are implemented separately. During Top level STA analysis, SPEFs of these blocks are stitched together with top for flat STA analysis. This works well if pin location of blocks is not required. In case of AOCV analysis, when spatial analysis is enabled, accurate location of all nodes (pin/net) in the design is required for calculating accurate derates in the design. This makes it necessary to determine accurate nodes location during SPEF reading. This can be achieved by transforming SPEF coordinates as per their physical locations.

Note:

- This command works with all possible SPEF and RCDB input combinations with `read_parasitics` command, which means for only RCDB(s) as input, for a mix of SPEF(s) and RCDB(s) as input, and for only SPEF(s) as input.
- If this command is not specified, the `read_spef` command tries to automatically find the orientation and origin of the hierarchical SPEF blocks and applies those identified transforms to the parasitic co-ordinates.
- The `create_parasitic_coordinate_transform` command settings remain in effect for all the subsequent `read_spef` and `read_parasitics` until they are reset using the `-reset` option.
- Whenever the block's origin (in block SPEF/DEF) is non-zero, the x and y coordinates need to be specified for applying the transform. The last two parameters, `-x_block_origin` and `-y_block_origin` in the following command use-model are only required to be specified for the case of non-zero block origin.

  ```
  create_parasitic_coordinate_transform -inst inst_name
  ```

```
-x_offset x-offset of block origin in top-level DEF/SPEF
-y_offset y-offset of block origin in top-level DEF/SPEF
-orient orientation
-x_block_origin x-coord of non-zero block origin
-y_block_origin y-coord of non-zero block origin
```

For example:

- **For Non-Zero Origin Block**: Specify the block origin as shown below:

  ```
  create_parasitic_coordinate_transform -inst eco_inst1
  -x_offset 421.6
  -y_offset 485
  -orient R180
  -x_block_origin 3.4
  -y_block_origin 9.7
  ```

- **For zero origin block**: No need to specify the block origin as shown below:

  ```
  create_parasitic_coordinate_transform
  -inst controller
  -x_offset 16.8
  -y_offset 205
  -orient R0
  ```

  or

  ```
  create_parasitic_coordinate_transform
  -inst controller
  -x_offset 16.8
  -y_offset 205
  -orient R0
  -x_block_origin 0
  -y_block_origin 0
  ```

  Both the above commands are equivalent.

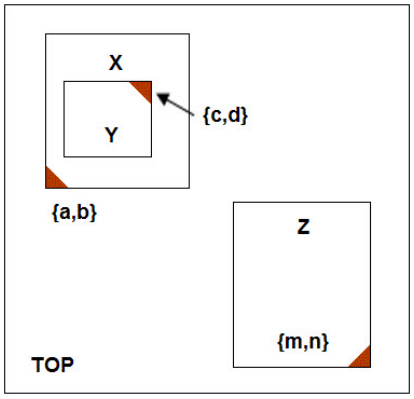
- The impact of this command is valid till first `read_spef` command is read (in case of MMMC, till all corners SPEFs have been read). If you want to read SPEF again, this command needs to be applied once again. These transformations will be applied during `read_spef` command.

Parameters

`-help`	Prints out the command usage. For a detailed description of the command and all of its parameters, use the `man` command: `man create_parasitic_coordinate_transform`.
`-inst` *`inst_name`*	Indicates the hierarchical name of the instance for which SPEF is being read. *Data_type*: string, required
`-orient {R0 R90 R180 R270 MY MX90 MX MY90}`	Indicates orientation change of instances. The valid values are `R0`, `R90`, `R180`, `R270`, `MX`,`MX90`, `MY`, and `MY90`. *Data_type*: enum, required
`-relative_to_parent`	When this parameter is specified, the software considers block offsets for the specified instance relative to the immediate parent. By default, the software considers the offset value relative to the top hinst.
`-reset`	Resets SPEF transform values. *Data_type*: bool, required
`-x_block_origin` *`value`*	Specifies that in case of non-origin block, offset x-coordinate in microns. *Data_type*: float, optional
`-x_offset` *`value`*	Indicates the actual x-coordinate as exists in the physical floor plan, corresponding to the specified instance. *Data_type*: float, required
`-y_block_origin` *`value`*	Specifies that in case of non-origin block, offset y-coordinate in microns *Data_type*: float, optional
`-y_offset` *`value`*	Indicates the actual y-coordinate as exists in the physical floor plan, corresponding to specified instance. *Data_type*: float, required

Examples

Consider the following hierarchical design where X and Z are two blocks inside TOP, and Y is another hierarchical block in X:



Here, X,Y, and Z have the following physical locations in the floor plan:

```
X - (a,b)
Y- (c,d)
Z- (m,n)
```

You can apply transformations to all blocks with respect to TOP, by using the following commands:

```
create_parasitic_coordinate_transform -inst X -x_offset a -y_offset b

create_parasitic_coordinate_transform -inst Z -x_offset m -y_offset n -orient MY

create_parasitic_coordinate_transform -inst X/Y -x_offset c -y_offset d -orient R180
```

These transformations will be applied when `read_spef` command is issued.

Related Commands

- set_analysis_view
- redirect
- read_parasitics
- read_spef

extract_rc

```
extract_rc
[-help]
```

Extracts resistance and capacitance for the interconnects and stores the results in an RC database. Optionally, an ASCII report file can be generated. Use this command after performing trial routing or final routing, and after setting the extract_rc attributes.

Parameter

`-help`	Outputs a brief description of the `extract_rc` command. For a detailed description of the command, use the man command: `man extract_rc`.

Example

The following command extracts interconnect resistance and capacitance:

```
extract_rc
```

RC database is created.

The `extract_rc` command can be followed by the `write_parasitics` command to extract the parasitics from the RC database, and to generate an ASCII report containing the database information.

Related Information

- extract_rc Category Attributes chapter in the *Innovus Stylus Common UI Text Command Reference*
- RC Extraction chapter in the *Innovus Stylus Common UI User Guide*

extract_rc Category Attributes

The root attributes in the `extract_rc` category are:

- extract_rc_assume_metal_fill
- extract_rc_cap_filter_mode
- extract_rc_cerebrus_license_only
- extract_rc_compress_rcdb
- extract_rc_coupled
- extract_rc_coupling_cap_threshold
- extract_rc_def_via_cap
- extract_rc_effort_level
- extract_rc_engine
- extract_rc_extra_cmd_file
- extract_rc_hard_block_obs
- extract_rc_incremental
- extract_rc_layer_independent
- extract_rc_lef_tech_file_map
- extract_rc_local_cpu
- extract_rc_pvs_fill
- extract_rc_qrc_cmd_file
- extract_rc_qrc_cmd_type
- extract_rc_qrc_output_mode
- extract_rc_qrc_run_mode
- extract_rc_qrc_stream_map_file
- extract_rc_relative_cap_threshold
- extract_rc_shrink_factor
- extract_rc_signoff_stream_layer_map
- extract_rc_total_cap_threshold
- extract_rc_tquantus_model_file
- extract_rc_tsv_subckt_file
- extract_rc_turbo_reduce
- extract_rc_use_qrc_oa_interface
- extract_rc_use_shielding_in_detail_mode
- extract_rc_via_cap

To set the value of any root attribute, use the `set_db` command:

```
set_db attribute_name value
```

To get the current value of any root attribute, use the `get_db` command:

```
get_db attribute_name
```

To get the current value of all attributes in the `extract_rc` category, use the following command:

```
get_db -category extract_rc
```

extract_rc_assume_metal_fill

```
extract_rc_assume_metal_fill scalevalue
```

Default: `0`

Data_type: double, read/write

Takes into account the metal fill in the design. For post-Route extraction, only extraction engine with `extract_rc_effort_level low` supports this attribute.

In post-Route extraction with extract_rc_effort_level low, scalevalue is not required and will be ignored. Detail extraction uses the FILLACTIVESPACING values from LEF to account for metal fill in the design. If the FILLACTIVESPACING values are not specified in the LEF file, the software uses the default value, which is 0.6um.

In pre-Route extraction, when calculating the capacitance of a wire segment in layer N, this attribute always assumes layer N-1 and layer N+1 to be filled. Specify one of the following values:

- 0: Assume no metal fill in layer N. When you do not specify this attribute, the software assumes no metal fill in layer N. This is equivalent to setting the scalevalue option to 0.
- 1: Assume full metal fill at minimum spacing distance. Where there is a minimum of signal-to-signal spacing, the cross-coupling capacitance will be extracted and reported. Where there is no wire within the minimum spacing, a capacitance to ground will mimic minimum-spaced metal fill in that location.
- x: Capacitance value is calculated once with full metal fill and once without metal fill. The user-specified scale value (decimal value between 0 and 1) determines the interpolation point between the two calculated values, for example, 0.5.

Related Command

- `extract_rc`

extract_rc_cap_filter_mode

```
extract_rc_cap_filter_mode { relative_only | relative_and_coupling | relative_or_coupling }
```

Default: For TQuantus, IQuantus, and Standalone Quantus extraction, the default value is `relative_and_coupling`. For post-Route extraction with `extract_rc_effort_level low`, the default value is dependent on the process node specified with the `init_process_node` attribute. If the process node is 130nm or below, the default value is `relative_and_coupling` and if the process node is above `130nm`, the default value is `relative_only`.

Data_type: enum, read/write

Specifies the capacitance filtering mode. The filtering modes are conditions which define how to use the `extract_rc_coupling_cap_threshold` and `extract_rc_relative_cap_threshold` values, or their combination. Based on the filtering mode specified, the software determines the nets in the design, which will have their coupling capacitance lumped to the ground.

Note: The behavior of the extract_rc_total_cap_threshold attribute is independent of the `extract_rc_cap_filter_mode` attribute setting. Nets that have a total capacitance value lower than the value specified with the `extract_rc_total_cap_threshold` attribute will have their coupling capacitances grounded regardless of the `extract_rc_cap_filter_mode` attribute setting.

You can use one of the following modes:

- `relative_only`: Grounds the coupling capacitance of nets based on the threshold value set by the `extract_rc_relative_cap_threshold` attribute.
- `relative_and_coupling`: Grounds the coupling capacitance of nets only when it is lower than the threshold value specified with the `extract_rc_relative_cap_threshold` attribute as well as the `extract_rc_coupling_cap_threshold` attribute. In this mode, you enforce maximum restriction on grounding the coupling capacitances.
- `relative_or_coupling`: Grounds the coupling capacitance of nets if it is lower than the threshold value specified with one of the `extract_rc_relative_cap_threshold` or `extract_rc_coupling_cap_threshold` attributes.

Related Command

- `extract_rc`

extract_rc_cerebrus_license_only

```
extract_rc_cerebrus_license_only {true | false}
```

Default: `false`

Data_type: bool, read/write

This attribute is applicable to the Cerebrus-Quantus flow.

When set to `true`, only the Cerebrus license will be considered and Innovus will not fall back to use Quantus (QTS) license when Cerebrus license(s) is/are not available. By default, the attribute is set to `false`, which means the regular QTS license will be considered when Cerebrus license(s) is/are not available.

extract_rc_compress_rcdb

```
extract_rc_compress_rcdb {true | false}
```

Default: false

Data_type: bool, read/write

Generates compressed RCDB data.

Related Command

- extract_rc

extract_rc_coupled

```
extract_rc_coupled {true | false}
```

Default: true (for post-Route extraction)

Data_type: bool, read/write

Separately outputs grounded and coupling capacitance. The false value of this attribute is typically used for static timing analysis. Signal Integrity analysis requires the coupled mode.

Related Command

- extract_rc

extract_rc_coupling_cap_threshold

```
extract_rc_coupling_cap_threshold file_name
```

Default: 0.2fF

Data_type: double, read/write

Specifies the threshold value that determines when the extractor lumps a net's coupling capacitance to ground. This attribute is applicable only when the `extract_rc_coupled` attribute is set to 1. The software decouples the coupling capacitance of nets when the total coupling capacitance between the pair of nets is lower than the threshold specified with this attribute. The actual decoupling also depends on the setting of the `extract_rc_cap_filter_mode` attribute.

If the `init_process_node` attribute is used, the software assigns the recommended default value for this attribute based on the process node specified.

Note: See the `init_process_node` attribute for the list of default values that are applied based on the process node setting.

Related Command

- extract_rc

extract_rc_def_via_cap

```
extract_rc_def_via_cap {true | false}
```

Default: `false`

Data_type: bool, read/write

Takes into account the via capacitance in the pre-Route RC extraction results. Specify a value of `true`, if you are experiencing correlation issues caused by deviations in clock latency when running pre-Route RC extraction.

Note: For 20nm and below nodes, the default value of this attribute is set to `true`.

Related Command

- extract_rc

extract_rc_effort_level

```
extract_rc_effort_level {low | medium | high | signoff}
```

Default: `low or medium` (depending upon the process node.)

Default: `undefined` for pre-Route extraction

Data_type: enum, read/write

Specifies the post-Route engine variant that you want to use for performing RC extraction.

Note: TQuantus (`extract_rc_effort_level medium`) is the default extraction engine in the post-Route flow for 65 nm and below design. However, TQuantus and IQuantus require a Quantus Techfile. Therefore, Native Detailed (`extract_rc_effort_level low`) engine remains the default engine if no Quantus Techfile has been defined.

This attribute supports the following values:

- `low`: Invokes the native detailed extraction engine. This is the same as specifying the `extract_rc_engine post_route` setting.
- `medium`: Invokes the Turbo-Quantus (TQuantus) engine. TQuantus performance and accuracy falls between native detailed extraction and IQuantus engine. This engine supports distributed processing.
  TQuantus engine is recommended for process nodes < 65nm.
  Note: This setting does not require a Quantus license.
- `high`: Invokes the Integrated-Quantus (IQuantus) extraction engine.
  IQuantus provides superior accuracy compared to TQuantus. IQuantus is recommended for extraction after ECO. In addition, IQuantus supports distributed processing.
  Note: IQuantus requires a Quantus-XL license.
- `signoff`: Invokes the Standalone Quantus extraction engine. This engine choice provides the highest accuracy. The engine has several runModes, thereby, providing maximum flexibility.

Related Command

- `extract_rc`

extract_rc_engine

```
extract_rc_engine {pre_route | post_route}
```

Default: `pre_route`

Data_type: enum, read/write

Specifies the extraction engine to use.

- `pre_route`: Uses pre-Route engine. RC extraction is done by the fast density measurements of the surrounding wires; coupling is not reported. Use this mode to perform extraction on pre-routed designs after clock tree synthesis.
- `post_route`: Uses post-Route engine. RC extraction is done by the detailed measurement of the distance to the surrounding wires; coupling is reported. The `extract_rc_effort_level` attribute further specifies which post-Route engine is used for balancing performance versus accuracy needs.

Related Command

- `extract_rc`

extract_rc_extra_cmd_file

```
extract_rc_extra_cmd_file filename
```

Default: `""` (empty string)

Data_type: string, read/write

Specifies the Quantus extra command file for TQuantus/IQuantus extraction. The extra command file supports the following variables:

- `log_file -dump_options { true | false}`
- `extraction_setup -copy_port_to_obs { true | false}`

- `extraction_setup -remove_net_pin_overlap {true | **false** }`
- `metal_fill -type {grounded | **floating** | virtual}`
- `output_setup -compress_temporary_files {true | **false** }`

Notes:

- TQuantus only supports advanced virtual metal fill in `metal_fill` in the extra command file. It does not support the grounded and floating options.
- When the `metal_fill` type is virtual, provide the absolute paths to the vmf rule file and the metal schema file. For example,

```
metal_fill \
   -type                                virtual \
   -enable_advanced_virtual_fill            true \
   -vmf_metal_scheme_file .../saveTestcase_vmf/missing_vmf_files/abc.vmf_metal_scheme \
   -vmf_rule_file .../saveTestcase_vmf/missing_vmf_files/abc.vmf_rule
```

- The `extract_rc_extra_cmd_file` attribute becomes “non-portable” if the vmf rule file and/or schema files have been specified. This means that the two file paths will not be automatically modified or ported. You can modify the file paths, if required. To make the extra command file portable, ensure that the vmf rule file and schema files are available at the locations specified in the extra command file.
- The **bold** text in the above example indicates that these values are used by default when the commands are not explicitly specified in the extra command file. For example, IQuantus extracts the “**`floating`**” metal fills by default when either the `-extraCmdFile` option is not specified or the `metal_fill` variable is not specified in the extra command file.
- This parameter is not available on the IBM platform.

Related Command

- `extract_rc`

extract_rc_hard_block_obs

```
extract_rc_hard_block_obs {true | false}
```

Default: `false`

Data_type: bool, read/write

Makes cell geometries on metal layers from obstruction, and power and ground pin shapes visible to the extractor for hard blocks only. The software recognizes the following three types of hard blocks:

- Hard macro (CLASS BLOCK in LEF file)
- Blackbox (CLASS BLACKBOX in LEF file)
- Partition (Committed during Innovus session)

Specify a value of 1 when routing goes over the listed hard blocks. A value of true removes the optimism for such routes, and provides better correlation with sign-off extraction without significantly impacting runtime.

Related Command

- `extract_rc`

extract_rc_incremental

```
extract_rc_incremental {true | false}
```

Default: true

Data_type: bool, read/write

Uses incremental IQuantus extraction mode. If the incremental mode is set after performing initial extraction on full design, extraction checks for design changes, and if the amount of changes are below the threshold, extraction is rerun in incremental mode on the modified areas only. Incremental mode can provide significant performance improvement during the signal integrity optimization phase.

Related Command

- `extract_rc`

extract_rc_layer_independent

```
extract_rc_layer_independent {true | false}
```

Default: `true`

Data_type: int/read/write

By default, this attribute is set to `true` or `1`. In this mode, layer-independent extraction is enabled. This mode of extraction is recommended for designs with layer stacks that have significant variations in RC characteristics. This mode uses advanced techniques to avoid timing jumps due to differences in routing-layer assignments between pre-Route and post-Route stages.

When this attribute is set to `false` or `0`, layer-independent extraction is disabled.

Note: pre-Route extraction scale-factors should be computed without setting this mode.

Related Command

- `extract_rc`

extract_rc_lef_tech_file_map

```
extract_rc_lef_tech_file_map filename
```

Default: "" (empty string)

Data_type: string, read/write

Specifies the location of the optional layer map file for the IQuantus and Standalone Quantus engines.

Note: For IQuantus, layer mapping between the LEF or DEF file and the Quantus technology file is done automatically. This attribute is only necessary if this mechanism fails. If necessary, you can use a copy of the automatically generated layer map file `extLogDir/IQRC_*/<design>.layermap.log` and modify it for your own use. Layer map file is required for use with Standalone Quantus. For TSV designs, you are required to provide the name of the layer map file for IQuantus and Standalone Quantus.

Note: If used, the layer map file must be in the Common Command Language (CCL) format. The following is an example of the layer map file format:

```
extraction_setup \
-technology_layer_map \
# <LEF/DEF Name> #<Tech File Name>
M1 METAL_1 \
VIA1 VIA_1 \
M2 METAL_2 \
```

Note: For cell-level IQuantus layer map, the DEF/LEF layer names are case sensitive, but ICT layer names are case-insensitive.

Note: TQuantus is not impacted by this attribute because it always automatically generates the layer map.

Related Command

- `extract_rc`

extract_rc_local_cpu

```
extract_rc_local_cpu number_of_cpus
```

Default: As specified in the `set_multi_cpu_usage -local_cpu` setting.

Data_type: int, read/write

Specifies the number of CPUs to be used for IQuantus extraction. This attribute is only applicable for the IQuantus extraction engine. When specified, it overrides the `set_multi_cpu_usage -local_cpu` option for IQuantus.

Related Command

- `extract_rc`

extract_rc_pvs_fill

`extract_rc_pvs_fill {true | false}`

Default: `false`

Data_type: bool, read/write

When this attribute is set to true, the PVS fill data attached to the Innovus database is sent to Quantus so that the effects of the fill data are extracted. This attribute only affects signoff Quantus extraction. It has no effect on tQuantus or iQuantus extraction done inside Stylus. This atttribute requires that either the `extract_rc_qrc_stream_map_file` attribute or the `extract_rc_signoff_stream_layer_map` attribute is specified.

Related Command

- `extract_rc`

extract_rc_qrc_cmd_file

`extract_rc_qrc_cmd_file` *`filename`*

Default: "" (empty string)

Data_type: string, read/write

Specifies the user created Quantus command file to be used for running Standalone Quantus. This command file should be in the new CCL syntax.

Note: This attribute is only valid if `extract_rc_effort_level signoff` and `extract_rc_qrc_cmd_type partial` or `custom` is selected.

Related Command

- `extract_rc`

extract_rc_qrc_cmd_type

`extract_rc_qrc_cmd_type {auto | partial | custom}`

Default: auto

Data_type: enum, read/write

Specifies whether to derive the settings for running Quantus from the software settings, or from user created command file using the `extract_rc_qrc_cmd_file` attribute.

The `extract_rc_qrc_cmd_type` attribute is only valid if `extract_rc_effort_level signoff` is selected. This attribute supports the following values:

- `auto`: Derives the Quantus settings automatically from the software parameter settings.
  Note: In this mode, the extract_rc_lef_tech_file_map attribute is required to be specified.
- `partial`: Appends user commands from the qrcCmdFile to the settings derived from the software parameter settings. You must provide the partial mode command file in CCL syntax. Use the partial command file only when you add extraction directives for those options that cannot be set in software. Following are some important restrictions:
  - Do not specify the following options in the partial mode command file:
    - Threshold setting
    - Coupled/decoupled mode
    - Quantus Techfile (s)
    - Operating Temperature
    - Do not specify "`-directory_name`" and "`-temporary_directory_name`" and `"-file_name"` in `output_setup` section of the command file
  - Do not specify output file using `output_setup -file_name` command in partial command file
    Note: The multi-temperature multi-corner extraction (MTMC) feature is enabled in the partial mode only if process technology is not specified in the user command file. In MTMC extraction, for RC corners with same Quantus technology file but different temperatures, resistance is extracted for

multiple corners while capacitance is extracted only for a single corner.
Note: There is no checking for syntax or content of this file, and it will be appended as is to the automatically created Quantus command file.

- `custom`: Gets the complete Quantus settings from the user created command file. Except for input and output, no settings from software parameters are used. You must provide the complete Quantus command file in CCL syntax.
  Following are some important restrictions:
  - Do not specify an input DEF/OA DB in the custom mode command file
  - Do not specify "`-directory_name`" and "`-temporary_directory_name`" in `output_setup` section of the command file.
  - Specify output SPEF with the following convention for the name:`<design.spef>`.
    In MMMC, the output SPEF files will be named: `<design>_<corner>_<Temperature>.spef`. Here "design" refers to the top cell name.
    Note: Do not prefix or postfix the output file name.
  - Specify the Quantus Techfile(s) to be used. In MMMC, the corner names should match the names used in the `create_rc_corner` command
  - Provide Quantus Techfile layer mapping information in custom command file
  - If GDS is provided for cell contents, provide a Quantus Techfile to GDS layer number matching

Note: The MTMC extraction feature is not enabled in the custom mode of Quantus extraction.

Note: There is no checking for syntax or content of the command file prior to running the extraction. The command file will be used as is. If the output file is not specified correctly, the resulting parasitic file will not be read into the Stylus RC database.

Related Command

- `extract_rc`

extract_rc_qrc_output_mode

```
extract_rc_qrc_output_mode {rcdb | spef}
```

Default: `spef`

Data_type: enum, read/write

Sets the Quantus output write mode to RCDB or SPEF.

Note: This attribute is only valid if `extract_rc_effort_level` is `signoff`.

Related Command

- `extract_rc`

extract_rc_qrc_run_mode

```
extract_rc_qrc_run_mode {concurrent | sequential}
```

Default: `concurrent`

Data_type: enum, read/write

Specifies the Quantus run mode in MMMC designs.

Note: For MMMC designs, in Quantus partial command type, do not specify the output SPEF file name and RC corner name in the command file to run Quantus in concurrent or sequential mode. For MMMC designs, in Quantus custom command type, do not specify the output SPEF file name and RC corner name together in the command file. Quantus sequential mode is not supported in custom command type.

This attribute supports the following values:

- `concurrent`: Runs Quantus for all active corners concurrently.
- `sequential`: Runs Quantus for all active corners sequentially.

Related Command

- `extract_rc`

extract_rc_qrc_stream_map_file

`extract_rc_qrc_stream_map_file` *`filename`*

Default: `""` (empty string)

Data_type: string, read/write

This attribute is specified only when `extract_rc_pvs_fill` attribute is set to true. When the `extract_rc_pvs_fill` attribute is set to `true`, a Quantus stream/oasis map Tcl command is required to map the PVS layer/dtype data to the corresponding Quantus layers.

If the `pvs_fill.data` is generated based on (GDS), the map file should be in the following format:

```
#layer gds_layer gds_dtype
<layer_name> <layer_number> <data_type>
…
```

If the `pvs_fill.data` is generated based on OASIS, the map file should be in the following format:

```
#layer oasis_layer oasis_dtype
<layer_name> <layer_number> <data_type>
…
```

Where,

- `layer_name` is the layername in the design
- `layer_number` is the layer ID in the GDS/OASIS file
- `data_type` is the datatype specified in the GDS/OASIS file

For example,

```
M1  31  0
M1  31  1
M2  32  0
M2  32  1
```

Note: This attribute only affects signoff Quantus extraction.

Related Command

- `extract_rc`

extract_rc_relative_cap_threshold

`extract_rc_relative_cap_threshold` *`value`*

Default:`1.0`

Data_type: double, read/write

Sets a ratio threshold value that determines when the extractor lumps a net's coupling capacitance to ground. This attribute is applicable only when the `extract_rc_coupled` attribute is set to 1. If the total coupling capacitance between a pair of nets is less than the percentage (specified with this attribute) of the total capacitance of the net with the smaller total capacitance in the pair, the coupling capacitance between these two nets will be considered for grounding. The actual decoupling also depends on the setting of the `extract_rc_cap_filter_mode` attribute.

The range of this value is 0 to 1. If the `init_process_node` attribute is used, the software assigns the recommended default value for this attribute based on the process node specified.

Note: See the `init_process_node` attribute for the list of default values that are applied based on the process node settings.

Related Command

- `extract_rc`

extract_rc_shrink_factor

```
extract_rc_shrink_factor shrink_factor_value
```

Default: `1.0`

Data_type: double, read/write

Specifies the shrink factor by which to shrink the design before extraction. The shrink factor is used only for extraction calculations. It does not change the actual design. This attribute must only be used when restoring design.

Note: This attribute is not recommended for newer technologies where shrink factor is defined in the Quantus techfiles or capacitance tables.

Example:

- The following attribute sets the shrink factor to `0.9`:

  ```
  set_db extract_rc_shrink_factor 0.9
  ```

  Setting the shrink factor to `0.9` means the value of wire lengths, widths, and spacings in the design will shrink to 90% of their original values.

Related Commands

- `extract_rc`
- `set_extract_shrink_factor`

extract_rc_signoff_stream_layer_map

```
extract_rc_signoff_stream_layer_map file_name
```

Default: "" (empty string)

Data_type: string, read/write

This atrribute is specified only when the `extract_rc_pvs_fill` atrribute is set to `true`. When the `extract_rc_pvs_fill` atrribute is set to `true`, a Quantus stream/oasis map Tcl command is required to map the PVS layer/dtype data to the corresponding Quantus layers.

If the `pvs_fill.data` is generated based on stream (GDS/OASIS), the map file should be in the following format:

```
#layer gds_layer gds_dtype use-type color-ID
<layer_name> <layer_number> <data_type> <use-type> <color-ID>
```

…

Where,

- `layer_name` is the LEF layername in the design
- `layer_number` is the layer ID in the GDS/OASIS file
- `data_type` is the datatype of the layer in the GDS/OASIS file
- `use_type` is the keyword for identifying the use of the data. The valid values are:
  - `gray`: library data in the GDS/OASIS file
  - `fill`: regular metal fill data in the GDS/OASIS file
  - `active`: active metal fill data in the GDS/OASIS file
- `color-ID` is the color ID. Use "`1`" or "`2`" if the layer is colored, or use hyphen "`-`" if the layer is not colored

For example,

```
M1  31  0 gray -
M1  31  1 fill -
M2  32  0 gray -
M2  32  1 fill -
```

Note: This attribute only affects signoff Quantus extraction.

Related Command

- `extract_rc`

extract_rc_total_cap_threshold

```
extract_rc_total_cap_threshold value_in_fF
```

Default: `0fF`

Data_type: double, read/write

Specifies the threshold value (femtoFarads) that determines when the extractor lumps a net's coupling capacitance to ground. This attribute is applicable only when the `extract_rc_coupled` attribute is set to 1. The software grounds the coupling capacitances for the nets which have a total capacitance value less than the value specified with this attribute.

If the `init_process_node` attribute is used, the software assigns the recommended default value for this attribute based on the process node specified.

Note: See the `init_process_node` attribute for the list of default values that are applied based on the process node settings.

Related Command

- `extract_rc`

extract_rc_tquantus_model_file

```
extract_rc_tquantus_model_file filename
```

Default: "" (empty string)

Data_type: string, read/write

Specifies the name of the tQuantus model file to be used for post_route effort-level medium extraction. You can specify the model file as follows:

```
extract_rc_engine post_route
extract_rc_effort_level medium
extract_rc_tquantus_model_file tQuantus_model.bin
```

For details of the use model, see the RC Extraction chapter of the *Innovus Stylus common UI User Guide.*

Note : If this file is not specified, it will be generated automatically by the software.

Related Command

- `extract_rc`

extract_rc_tsv_subckt_file

```
extract_rc_tsv_subckt_file filename
```

Default: "" (empty string)

Data_type: string, read/write

Specifies the name of the Through Silicon Via (TSV) subckt file to be used. The file contains the .subckt model information for TSV. The file has a Spice format. A sample file is shown below.

```
.subckt TSV top bottom
R1 top n2 0.03075
R2 n2 bottom 0.03075
C1 n2 0 28f
.ends
```

Where,

`top` and `bottom` refer to the terminal nodes on the top and bottom layers of the TSV

`n2` is the internal node between the top and bottom layers

`R1 top n2 0.03075` means that a resistor with a resistance of `0.03075 Ohms` is added between the `top` and `n2` nodes

`R2 n2 bottom 0.03075` means that a resistor with a resistance of `0.03075 Ohms` is added between the `bottom` and `n2` nodes

`C1 n2 0 28f` means that a ground capacitance of `28 fF` is added on the `n2` node

Note: The IQuantus engine might not keep the subckt connectivity as it is. The optimization of RES network will remove node n2 (if there is no other connections) and keep a single resistor between the top and bottom nodes with an accumulated resistance. For example, in the above example, the accumulated resistance will be `0.0615 Ohms`. Capacitance will also be moved to a "higher layered node". In the above example, this layer is `top`.

Related Command

- `extract_rc`

extract_rc_turbo_reduce

```
extract_rc_turbo_reduce {true | false | auto}
```

Default: `auto`

Data_type: enum, read/write

Controls the turbo reduction feature for TQuantus/IQuantus, and Standalone Quantus extraction. You can set the following options:

- `auto`: The software automatically turns `on` turbo reduction and proceeds when the required Quantus license is available. However, if the required Quantus license is not installed in the license server, the software automatically turns off Turbo reduction and proceeds with a warning message. If the license is installed in the license server but temporarily not available, then two scenarios are possible. First, if the license wait mode is not turned on, then Quantus will look for the license right away and error out because the license is not available. Second, if the license wait mode is turned on, then IQuantus will wait and check for the license periodically until timeout. If the license is available before the timeout, turbo reduction will be performed but if the license is not available before timeout, IQuantus will error out.
- `true`: The turbo reduction engine is `on`. However, if the required Quantus license is not installed in the license server, then Standalone Quantus and IQuantus will error out with a message and the run will stop. It is because setting the option to true means the user really wants to perform turbo reduction.
- `false`: The turbo reduction engine is switched `off`.

For details, see the table below. However, note that the licensing rules given in this table are not applicable to TQuantus because TQuantus does not require any Quantus license.

Behavior of turbo Reduction

	Availability of Quantus QRC AA License		
Option Selected – extract_rc_turbo_reduce{1 | 0 | auto}	Quantus QRC AA Installed and Available (IQRC gets the license right away even in the wait mode because sufficient licenses are available in the license server)	Quantus QRC AA Installed but not Available (If license wait mode is not turned on, then Quantus looks for the license . If the wait mode is turned on and if the license is temporarily not available, then IQRC waits and checks for the license periodically until timeout.)	Quantus QRC AA Not Installed on License Server
auto	perform turbo reduction	1. error out if wait mode is not on or if the specified wait time is exceeded 2. perform turbo reduction if wait mode is on and the license is available within the specified time.	print warning and do not perform turbo reduction
1	perform turbo reduction	error out	error out because the user explicitly wants to use turbo reduction
0	do not perform turbo reduction	do not perform turbo reduction	do not perform turbo reduction

Related Command

- `extract_rc`

extract_rc_use_qrc_oa_interface

```
extract_rc_use_qrc_oa_interface {true | false}
```

Default: `false`

Data_type: bool, read/write

Specifies that OpenAccess (OA) interface be used for invoking Quantus extraction instead of using the DEF-based flow.

Related Command

- extract_rc

extract_rc_use_shielding_in_detail_mode

```
extract_rc_use_shielding_in_detail_mode {true | false}
```

Default: false

Data_type: bool, read/write

Assumes shielding for any wire segment of a clock net that does not have any shielding yet, and the space between the wire segment and its neighbor is greater than or equal to the following:

```
2s+w
```

where,

`s` = minimum spacing or non-default spacing

`w` = minimum width or non-default width

This attribute is only available in post-Route mode when `extract_rc_effort_level` attribute is set to `low`.

Related Command

- extract_rc

extract_rc_via_cap

```
extract_rc_via_cap {false | true}
```

Default: true

Data_type: bool, read/write

Extracts capacitance values considering part of the via geometries that do not overlap with the wire geometries. Via geometries that do not overlap the wire geometries are often found if you are designing using newer design technologies such as 90nm or when using multiple cut via.

This attribute is only available in post-Route mode when `extract_rc_effort_level` attribute is set to `low`.

Related Command

- extract_rc

read_parasitics

```
read_parasitics
[-help]
[-force]
[-keep_star_node_locations]
[-rc_corner_map rc_corner1:spef_field1 rc_corner2:spef_field2 ...]
{[{{[-rc_corner <string>]}}]}
[-via_variation_file viaVariationFile]
```

Reads SPEF and RCDB-based RC parasitics information and already created RCDBs in Innovus. For hierarchical designs, use this command to read top-level as well as block-level parasitic information.

Note: The following are important aspects to be considered for using this command:

- The RCDB parasitic file must have the `.rcdb.d` extension.
- The RCDB-based parasitic data that is provided as input to this command must be the one that was saved in the same major release of the software. The command will not read the RCDB data saved in a previous release. For example, if you provide RCDB saved in the 10.1.x release of the software as input to the `read_parasitic`s command in the 11 software release, then it cannot be read.

- Only the RCDB parasitic data on systems that are same as those used for saving the data can be read. For example, if the data is saved on Linux in 64-bit mode, you can only restore it on Linux 64-bit mode.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `read_parasitics` parameter. For a detailed description of the command and all of its parameters, use the man command: `man read_parasitics`.
`-force`	Allows reading of parasitic data even when block level RCDB/ SPEF is missing for a corner. **Note:** This parameter is only applicable for hierarchical designs. It is required that the top cell should have parasitic data specified for all the active RC corners. *Data_type*: bool, optional
`-keep_star_node_locations`	Specifies whether or not to read the *N statements in the SPEF file. The *N statements define the location of the RC nodes. When this parameter is specified, the *N statements are read and when it is not specified, the *N statements are ignored. *Default*: `true` *Data_type*: bool, optional
`-rc_corner_map` *rc_corner1:spef_field1 rc_corner2:spef_field2 ...*	
	Provides a mapping of the RC corners to the SPEF fields of multi-value SPEF files. *Data_type*: string, optional
`-rc_corner` *list of RCcorner specific RCDBs/SPEFs for hierarchical design | RCcorner specific RCDB/SPEF*	
	Specifies the RC corner name along with a corner-specific RCDB or SPEF for both top and block levels. For hierarchical designs, this parameter lists the corner-specific SPEFs/RCDBs for both top and block levels. **Note:** A corner-specific RCDB parasitic file can only be provided as input to the RC corner for which it was generated. It cannot be provided as input to other RC corners. *Data_type*: string, optional
`-via_variation_file` *viaVariationFile*	

	Specifies the name of the file for modeling via variation. The contents of the file include all the via layer names with the corresponding MEAN_SHIFT and STDDEV so that all the VIA resistance in SPEF can be mapped to their own variation multiplier. This option is used to specify the via_variation file only when the design has a single corner.
	For example,
	`read_parasitics`
	`-rc_corner rcCornerName \`
	`-via_variation_file` *viaVariationFileName*

Examples

- The example below is for hierarchical design with Top (T), Block 1(B1) and Block 2(B2), and two active RC corners, C1 and C2.
  - The following command specifies the RC corner names along with corner-specific RCDB and SPEF for both top and block levels:

```
read_parasitics \
-rc_corner C1 Top_C1.spef Block1_C1.spef Block2_C1.spef.gz \
-rc_corner C2 Top_C2.rcdb.d Block1_C2.spef Block2_C2.rcdb.d
```

Related information

- `extract_rc`
- `extract_rc Category Attributes`

read_rc

```
read_rc
[-help]
dir_name
```

Restores the RC extraction data that was previously saved either with the `write_db name -rc` or the `write_rcdb` command.

Note: The systems you use to save and restore RC extraction data must match (32-bit or 64-bit data models).

Parameters

`-help`	Outputs a brief description that includes type and default information for the `read_rc` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man read_rc`.
`dir_name`	Specifies the location of the RCDB directory. The syntax of the directory name is `*.rcdb.d`. For example, you can restore the parasitic data in `test.rcdb.d`. *Data_type*: string, required

Related Information

- `write_db`
- `write_rcdb`
- RC Extraction chapter in the *Innovus Stylus Common UI User Guide*.

read_spef

```
read_spef
[-help]
file_name_list
[-extended]
[-idx]
[-scale_nets_file string]
[-scale_rc]
[-keep_star_node_locations | -skip_star_node_locations]
[{-rc_corner list+of_rc_corners [-spef_field {1 2 3 … N}]}]
[-via_variation_file viaVariationFileName]
```

Loads resistors and capacitors for the interconnects in SPEF into the Innovus database to calculate delays or build a timing graph. Use this command to read in a SPEF file generated by a third-party extraction tool.

You can use this command to load a SPEF file, which is compressed (`.gzip`) format.

Note: This command supports multi-threaded SPEF reading scalable upto a maximum of four CPUs. The number of CPUs can be specified using the standard use model of Innovus.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `read_spef` parameter. For a detailed descripti parameters, use the man command: `man read_spef`.
`-extended`	Enables the reading of "`$llx=<value1> $lly=<value2> $urx=<value3> $ury=<value4> $LAYER=<number>`" to set the layer Quantus extended SPEF file. It also reads in extended `*LAYER_MAP section`, if it is present. This parameter can also read ir `<swidth> $lvl=<number>`" to set the width, silicon width, or layer for nodes of a resistance from StarRC extended SPEF file *Data_type*: bool, optional
file_name_list	Specifies a SPEF file or list of SPEF files as input for flat chip or hierarchical SPEF stitching respectively. *Data_type*: string, required
`-idx`	Enables Inter-Die Extraction (IDX) SPEF reading. IDX SPEF models the Inter-Die capacitive impact in the form of coupling die and IDX SPEFs need to be extracted by Quantus and in the extended SPEF (xSPEF) format. The `-extended` paramete read the xSPEF format. The software also supports multiple IDX SPEF reading. Flat die-level SPEFs need to be specified `idx` parameter, die-level and IDX SPEFs can be specified in any order.
`-keep_star_node_locations` `| -skip_star_node_locations`	Specifies whether or not to read the *N statements in the SPEF file. The *N statements define the location of the RC nodes The -`keep_star_node_locations` parameter reads the *N statements and the `-skip_star_node_locations` parameter ignore *Default:* `-keep_star_node_locations` (By default, the software reads the *N statements in the SPEF file.)

`-rc_corner` *`list_of_rc_corners`*	Annotates the parasitics to the predefined RC corners for multi-corner analysis. You can specify a list of RC corners to be a information when calculating delays in multi-mode multi-corner analysis (MMMC) mode, you must annotate the parasitics f A few examples are provided below to explain different scenarios of specifying rc corners are their corresponding corner in • `read_spef -rc_corner {rcCorner1 rcCorner2 rcCorner3 rcCorner4 rcCorner5} -spef_field {1 2 3 4 5} rc-spef.gz` This means that SPEF file can contain parasitics from N RC corners and names of RC corners in the RC corner list map the `-spef_field` option as shown above. Assuming that the SPEF file, `rc-spef.gz` has RC fields from the specified five would be interpreted as follows: 1st field in SPEF maps to RC corner, `rcCorner1` 2nd field in SPEF maps to RC corner, `rcCorner2` 3rd field in SPEF maps to RC corner, `rcCorner3` 4th field in SPEF maps to RC corner, `rcCorner4`, and 5th field in SPEF maps to RC corner, `rcCorner5` • `read_spef -rc_corner {rc_max rc_min} -spef_field {2 5} rc-spef.gz` Assuming `rc-spef.gz` in above example has RC fields from "N" RC corners, the above command would read SPEF da be interpreted as follows: 2nd field in SPEF maps to RC corner, rc_max 5th field in SPEF maps to RC corner, `rc_min` • `read_spef -rc_corner {rcCorner1 rcCorner2 rcCorner3} -spef_field {1 1 3} rc-spef.gz` Assuming `rc-spef.gz` in the above example has RC fields from N RC corners, the above command would read SPEF c interpreted as follows: 1st field in SPEF maps to RC corner, `rcCorner1` 1st field in SPEF also maps to RC corner, `rcCorner2` 3rd field in SPEF maps to RC corner, `rcCorner3` • `read_spef -rc_corner {rcCorner1 rcCorner2 rcCorner3 rcCorner4 rcCorner5} -spef_field {1 2 3 4 5} {top.spe block2.spef.gz}` In the above example, hierarchical design SPEF files are read in. It reads top level multi-field SPEF file followed by the from `block1` and `block2`. **Note:** SPEF reading will not be triggered until SPEF files for all active RC corners are specified. *Data_type*: string, optional
`-scale_nets_file` *`string`*	Enables the scaling of resistances and capacitances for user-specified nets as per the specified scale factors. The list of ne factors are specified using the `create_rc_corner` and `update_rc_corner` commands. **Note**: This parameter can only be used if the `-scale_rc` parameter is specified. Otherwise, it will be ignored. For example: `read_spef -scale_rc -scale_nets_file net_file_name -rc_corner r1 detail.spef` *Data_type*: string, optional
`-scale_rc`	Scales the SPEF read in for the specified corner as per the `post_route_cap`, `post_route_cross_cap`, and `post_route_res` s `create_rc_corner` and `update_rc_corner` commands. When this parameter is specified, the scale factors for extraction effort-level `high` are used for RC scaling. However, if they the scale factors for effort-level `medium` and `low`, in that order, if specified. **Note**: Scale factors of `post_route_clock_cap, post_route_cross_cap,` and `post_route_clock_res` are not applied even w *Data_type*: bool, optional

`-spef_field {1 2 3 … N}`	Specifies the list of RC corner indices in SPEF to be loaded. This parameter provides the flexibility to perform order-indepe to RC corners. For example, SPEF may contain data for 10 corner fields but you can choose to read data only for 6 corners one particular field to map to one or more RC corners. A few examples to illustrate different scenarios are provided in the d above. **Note:** This parameter is relevant only when the `-rc_corner` parameter is specified. The `-spef_field` is an optional parame assumes the default fields of {1 2 3 … N} where N is equal to the number of RC corners provided in the `-rc_corner` list. *Data_type*: string, required
`-via_variation_file` *viaVariationFileName*	

Specifies the name of the file for modeling via variation. The contents of the file include all the via layer names with corresp so that all the VIA resistance in SPEF can be mapped to their own variation multiplier. This option is used to specify the via has a single corner.

For example,

```
read_spef
    -rc_corner rcCornerName \

    -via_variation_file viaVariationFileName
```

A sample via_variation file is provided below:

```
// corner_name  : rcMax

// wildcard to cover default
object_type     : design
delay_type      : via

layer_name      : *

area            : 0.20 0.40 0.80

// no nominal for wildcard default
table_min_sigma : 0.80 0.90 0.95

table_max_sigma : 1.60 1.30 1.15

table_meanshift : 0.05 0.03 0.02

table_stdev     : 0.10 0.05 0.02

// per via layer data
object_type     : design
delay_type      : via
layer_name      : VIA1

Area            : 0.0004 0.0008 0.0016

table_min_sigma : 0.80 0.90 0.95

table_max_sigma : 1.60 1.30 1.15

table_meanshift : 0.015 0.015 0.012

table_stdev     : 0.214 0.150 0.117

object_type     : design

delay_type      : via

layer_name      : VIA2

area            : 0.0004 0.0008 0.0016

table_min_sigma : 0.80 0.90 0.95

table_max_sigma : 1.60 1.30 1.15

table_meanshift : 0.015 0.012 0.014

…..

……

table_stdev : 0.223 0.183 0.140
```

Note: Each line in the file lists the basic components of the statistical variation approach.

The values in the "area" line list all possible areas. In the above example, the area line lists three cases that can happen in of cases, for example one or four. The subsequent lines list the corresponding values matching to the areas defined in the a

Examples

- The following command loads the SPEF file `TOPCHIP_SP.spef` for a flat chip methodology for RC corner, `corner1`:
  ```
  read_spef -rc_corner corner1 TOPCHIP_SP.spef
  ```
- The following command loads multiple SPEF files for hierarchical SPEF stitching:
  ```
  read_spef -rc_corner corner1 { PTN/results_conv/results_conv.spef tdsp_core/tdsp_core/tdsp_core.spef PTN/TOPCHIP_SP/TOPCHIP_SP.spef }
  ```

- The following commands a) setup the appropriate scale factors for the corners `RCmax` and `RCmin` b) scale the SPEF read for the defined corners `RCmax` and `RCmin`:

```
=== MMMC setup ====

create_rc_corner -name RCmax -postRoute_cap 1.4 -postRoute_xcap 1.4 -postRoute_res 1.4
create_rc_corner -name RCmin -postRoute_cap 0.8 -postRoute_xcap 0.8 -postRoute_res 0.8
...

=== run script===
...

read_spef -rc_corner RCmax typSPEF.spef -scale_rc
read_spef -rc_corner RCmin typSPEF.spef -scale_rc
```

- The following command loads a SPEF file `pnta-spef.gz` that has RC fields from 5 RC corners:

```
read_spef -rc_corner {rcCorner1 rcCorner2 rcCorner3 rcCorner4 rcCorner5} -spef_field {1 2 3 4 5} pnta-spef.gz
```

  The above command will be interpreted as follows:

```
1st field in SPEF maps to RC corner rcCorner1
2nd field in SPEF maps to RC corner rcCorner2
3rd field in SPEF maps to RC corner rcCorner3
4th field in SPEF maps to RC corner rcCorner4
5th field in SPEF maps to RC corner rcCorner5
```

- The following command loads a SPEF file `pnta-spef.gz` that has RC fields from N RC corners:

```
read_spef -rc_corner {rc_corner1 rc_corner2} -spef_field {2 5} pnta-spef.gz
```

  The above command will read SPEF data for 2 RC corners only and will be interpreted as follows:

```
2nd field in SPEF maps to RC corner rc_corner1
5th field in SPEF maps to RC corner rc_corner2
```

- The following command loads a SPEF file `pnta-spef.gz` that has RC fields from N RC corners:

```
read_spef -rc_corner {rcCorner1 rcCorner2 rcCorner3} -spef_field {1 1 3} pnta-spef.gz
```

  The above command will read SPEF data for 3 RC corners only and will be interpreted as follows:

```
1st field in SPEF maps to RC corner rcCorner1
1st field in SPEF maps to RC corner rcCorner2 as well.
3rd field in SPEF maps to RC corner rcCorner3
```

- The following command loads hierarchical design SPEF files. The software reads the top level multi-field SPEF file followed by the multi-field SPEF file from `block1` and `block2`:

```
read_spef -rc_corner {rcCorner1 rcCorner2 rcCorner3 rcCorner4 rcCorner5} -spef_field {1 2 3 4 5} {top.spef.gz block1.spef.gz
block2.spef.gz}
```

Related Information

- `create_rc_corner`
- `update_rc_corner`
- RC Extraction chapter in the *Innovus Stylus Common UI User Guide*

report_net_parasitics

```
report_net_parasitics
[-help]
net_name
-rc_corner rc_corner_name
[-out_file output_file_name]
```

Reports the RC parasitics of the specified net.

The `report_net_parasitics` command reports a list of attackers in the order of worst coupling capacitance by default. The software determines and reports the probable top N worst attackers for a victim. The command output will show worst coupling capacitance, in addition to RC parasitics.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `report_net_parasitics` parameter. For a detailed description of the command and all of its parameters, use the man command: `man report_net_parasitics`.
net_name	Specifies the name of the net for which the RC parasitics are to be reported. *Data_type*: string, required
`-out_file` *output_file_name*	Specifies the name of the output file. *Data_type*: string, optional
`-rc_corner` *rc_corner_name*	Specifies the RC corner name for which the RC parasitics are to be reported. *Data_type*: string, required

Example

- The following command reports the RC parasitics for net, NET1 for RC corner, `rc_best`:

```
report_net_parasitics -rc_corner rc_best NET1

Net: NET1
RC Corner: rc_best
Total Cap = 0.473082 ff
Total Res = 2.104041 ohms
Total XCap = 0.334035 ff
Cap Distribution:

Total Ground Cap = 0.139046 ff
XCap With Nets:

NET3 = 0.059964 ff
NET4 = 0.040277 ff
NET5 = 0.233794 ff
```

Related Information

- `extract_rc`
- `extract_rc Category Attributes`
- RC Extraction chapter in the *Innovus Stylus Common UI User Guide*

report_rcdb

```
report_rcdb
[-help]
dir_name
```

Displays the contents of the RCDB being read.

Note:

1. This command is supported only for reading flat single RCDB. Multiple RCDB's cannot be read simultaneously using this command.
2. The RCDB being read must be generated on the same platform and architecture on which it is being read. For example, if you have generated the RCDB on Linux platform, then the `report_rcdb` command cannot report its content on the IBM AIX platform. Similarly, an RCDB generated on a 64-bit system cannot be reported on a 32-bit system.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `report_rcdb` parameter. For a detailed description of the command and all of its parameters, use the man command: `man report_rcdb`.
`dir_name`	Specifies the location of the RCDB directory. *Data_type*: string, required

Example

- The following command reports the contents of the RCDB being read in the `test.rcdb.d` directory:

```
report_rcdb test.rcdb.d
Opening parasitic data file 'test.rcdb.d/header.da' for reading.
RCDB VERSION           : 11.1
OS-bit                 : 64
OS-endianness          : little-endian
RCDB Compression       : on
RCDB Source            : QRC
DESIGN                 : test
RC-Corners:            : typical
Coupled                : yes
Node-loc               : yes
Statistical            : yes
```

Related Information

- `extract_rc`
- `extract_rc Category Attributes`
- RC Extraction chapter in the *Innovus Stylus Common UI User Guide*

report_rc_factors

```
report_rc_factors
[-help]
[-out_file file_name]
[-pre_route { true | false}]
[-post_route { low | medium }]
[-reference {low | medium | high | signoff | external_spef}]
[-spef_map_file file_name]
[-use_ostrich_filters file_name]
```

Generates the resistance and capacitance scale factors for optimal RC correlation by comparing the SPEF files generated by native extraction with user-specified reference RC data or automatic running of extraction in higher accuracy mode. After generating the scale factors, the command prints a summary of the scale factors to a file. You can use this file to set the actual scale factors.

This command allows you to generate scale factors for one engine or multiple engines, depending on the parameters you specify. You can run pre-Route (`-pre_route`), or post-Route (`-post_route`) or both types of extraction to generate scale factors. For post-Route extraction, you can run extraction with effortLevel low or with effortLevel low and medium. The reference for comparison can be either an external SPEF or a SPEF generated from the post-Route extraction engine with the specified effort level.

When using an external SPEF as reference, use the `-spef_map_file` parameter to specify the name of a file containing multiple SPEF files for use in an MMMC setup.

Note: A Standalone Quantus extraction licence is required to run sign-off extraction in this command. Specify the Quantus path in the path setup. If you do not have a Standalone Quantus extraction license, you can use the `-spef_map_file` parameter to specify the signoff SPEF file(s). In addition, a proper setup of Quantus technology file(s) is required when specifying `-reference medium, high` or `signoff`.

Parameters

`-help`	Outputs a brief description that includes type and default information for the `report_rc_factors` parameter. For a detailed description of the command and all of its parameters, use the man command: `man report_rc_factors`.
`-out_file` *`file_name`*	
	Specifies the name of the file in which the command will save the new RC factor setting. You can use this file to set the actual scale factors. *Data_type*: string, optional
`-pre_route {true | false}`	
	Generates the scale factor for preRoute extraction. When set to `true`, generates optimal scale factors for both signal nets and clock nets for preRoute extraction. *Default*: `true` *Data_type*: enum, optional
`-post_route {low | medium}`	
	Specifies whether scale factors are to be generated for postRoute extraction. It also specifies whether scale factors are to be generated for engine with effortLevel low only, or for both engines with `effortLevel low` and `medium`. If the `-post_route` parameter is not specified, scale factors are not generated for postRoute extraction. When you use the `-post_route low` parameter, the software generates different scale factors for signal nets and clock nets. *Data_type*: enum, optional
`-reference {low | medium | high | signoff | external_spef}`	
	Specifies the postRoute engine or external SPEF that you want to use as reference for comparison. *Default*: `signoff` If you specify `external_spef`, you should use the `-spef_map_file` parameter to specify the third party SPEF file(s) to be used for comparison. *Data_type*: enum, optional
`-spef_map_file` *`file_name`*	

	Specifies the SPEF map file to be used for comparison when using MMMC designs. When you use this parameter, the design must be placed and fully routed, and the referenced SPEF file must be the one generated from the same routed design.
	The map file has a list of RC corners and the corresponding SPEF file for each RC corner, as shown in the following example:
	`rcmin design_rcmin.spef`
	`rcmax design_rcmax.spef`
	Data_type: string, optional

`-use_ostrich_filters file_name`	

	Specifies the filename containing user-specified ostrich filter values for Tcap, res, and Xcap. These user-specified values are applied during the Ostrich run. In case this parameter is not specified, default ostrich filter values will be used.
	The format of the file to be specified for ostrich filters is as follows:
	`-tcap {-min_data <value> -max_data <value> -min_fanout`
	`<value> -max_fanout <value> -max_perc_dev <value>}`
	`-res {-min_data <value> -max_data <value> -min_fanout <value>`
	`-max_fanout <value> -max_perc_dev <value>}`
	`-xcap {-min_data <value> -max_data <value> -min_fanout`
	`<value> -max_fanout <value> -max_perc_dev <value>}`
	Data_type: string, optional

Examples

- The following command generates the resistance and capacitance scale factors for the postRoute engine with `effortLevel low` in an MMMC design by comparing the parasitics with the SPEF file from signoff extraction:
  ```
  report_rc_factors -pre_route false -post_route low -reference signoff -out_file low_signoff.tcl
  ```

  Following are some sample sections in the log file for the above command:

  ```
  ====================================================

  Summary of scale factor generation

  ====================================================

  RC-Corner Name: rc_typical

  Extraction Engine "Postroute", Effort Level "low"

  Cap scale factor : 0.973457
  Res scale factor : 1.015816
  XCap scale factor : 0.91458
  Clock Cap scale factor : 1.031517
  Clock Res scale factor : 1.000277

  RC-Corner Name: rc_best

  Extraction Engine "Postroute", Effort Level "low"

  Cap scale factor : 1.087774
  Res scale factor : 0.843309
  XCap scale factor : 1.137418
  Clock Cap scale factor : 1.11112
  Clock Res scale factor : 0.827263

  RC-Corner Name: rc_worst

  Extraction Engine "Postroute", Effort Level "low"

  Cap scale factor : 0.92273
  Res scale factor : 1.186163
  XCap scale factor : 0.809296
  Clock Cap scale factor : 0.994961
  Clock Res scale factor : 1.170787

  Finished correlation ......................
  ```

- The following command generates scale factors for preroute and postroute -effortLevel low engines in an MMMC design by comparing the parasitics with an external supplied list of SPEF files:
  ```
  report_rc_factors -pre_route true -post_route low -reference external_spef -spef_map_file preRoute_mapFile
  ```

  In this example, the SPEF files specified in the file referenced in the `-spef_map_file` parameter must be generated using a postRoute extraction tool on the same routed design.

Related Information

- `extract_rc`
- extract_rc Category Attributes chapter of *Innovus Stylus Common UI Text Command Reference*
- RC Extraction chapter of *Innovus Stylus Common UI User Guide*

report_unit_parasitics

```
report_unit_parasitics
[-help]
-layer metal_layer_name
-rc_corner rc_corner_name
[-space wire_space_in_um]
[-via_res via_resistance]
[-width wire_width_in_um]
```

Reports unit parasitics for individual metal layers and vias. The report prints the per unit resistance and capacitance for the specified layer. Via parasitics are printed considering the layer mentioned in the command as the lower layer of the via. The default via specified in the LEF file will be considered in this case.

Note: The units for capacitance and resistance in the report are pF and ohms, respectively.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `report_unit_parasitics` parameter. For a detailed description of the command and all of its parameters, use the man command: `man report_unit_parasitics`.
`-layer` *metal_layer_name*	Specifies the name of the metal layer for which the unit parasitics are to be reported. *Data_type*: string, required
`-rc_corner` *rc_corner_name*	Specifies the name of the RC corner for which the unit parasitics are to be reported. *Data_type*: string, required
`-space` *wire_space_in_um*	Specifies the wiring space in microns. *Default*: minimum spacing *Data_type*: float, optional
`-via_res` *via_resistance*	When this parameter is specified, the resistances of all vias, whose upper layer is specified using the `-layer` parameter, are reported. *Data_type*: bool, optional
`-width` *wire_width_in_um*	Specifies the wiring width in microns. *Default*: minimum width *Data_type*: float, optional

Examples

- The following command reports the unit parasitics for metal layer, `Metal5`, for the specified RC corner, `rc_worst`:

  ```
  report_unit_parasitics -layer Metal5 -rc_corner rc_worst
  ```

  The software gives the following information:

  ```
  Unit parasitics for layer Metal5 (in fF,ohm)
      RC-Corner = rc_worst
        Cap = 0.213204
        Res = 0.900873
  Parasitics for default Via "VIA5_0_VH"
        Via Cap = 0.017901
        Via Res = 8.859576
  ```

- The following commands reports the unit parasitics for `Metal5` with spacing two times the minimum spacing:

  ```
  report_unit_parasitics -layer Metal5 -rc_corner rc_worst -space [expr 2 * [dbGet [dbGet head.layers.extName Metal5 -p].minSpacing]]
  ```

  The software gives the following information:

  ```
  Unit parasitics for layer Metal5 (in fF,ohm)

      RC-Corner = rc_worst

          Cap = 0.188657
          Res = 0.900873

  Parasitics for default Via "VIA5_0_VH"
  ```

```
        Via Cap = 0.017901
        Via Res = 8.859576
```

- The following commands reports the unit parasitics for Metal5 with width two times the minimum width:

  ```
  report_unit_parasitics -layer Metal5 -rc_corner rc_worst -width [expr 2 * [dbGet [dbGet head.layers.extName Metal5 -p].minWidth]]
  ```

  The software gives the following information:

  ```
  Unit parasitics for layer Metal5 (in fF,ohm)

      RC-Corner = rc_worst
      Cap = 0.198657
      Res = 0.555333

  Parasitics for default Via "VIA5_0_VH"

      Via Cap = 0.017901
      Via Res = 8.859576
  ```

- The following commands reports the unit parasitics for vias with top layer of Metal5:

  ```
  report_unit_parasitics -layer Metal5 -rc_corner rc_worst -via_res
  ```

  The software gives the following information:

  ```
  Unit parasitics for layer Metal5 (in fF,ohm)

      RC-Corner = rc_worst
      Cap = 0.213204
      Res = 0.900873

  Parasitics for default Via "VIA5_0_VH"

      Via Cap = 0.017901
      Via Res = 8.859576
      Other Via res
      VIA45_1C 8.859576
      VIA45_1C_H 8.859576
      VIA45_1C_V 8.859576
      VIA45_1ST_N 8.859576
      VIA45_1ST_S 8.859576
      VIA45_2C_E 4.429788
      VIA45_2C_W 4.429788
      VIA45_2C_N 4.429788
      VIA45_2C_S 4.429788
      VIA45_2ST_N 4.429788
      VIA45_2ST_S 4.429788
      VIA45_4C 2.214894
  ```

Related Information

- `extract_rc`
- `extract_rc Category Attributes`
- RC Extraction chapter of *Innovus Stylus Common UI User Guide*

reset_parasitics

```
reset_parasitics
[-help]
```

Deletes the parasitic database without resetting the extraction modes specified using the `extract_rc` category attributes.

Parameter

`-help`	Outputs a brief description of the `reset_parasitics` parameter.

Related Information

- `extract_rc`
- extract_rc Category Attributes chapter in the *Innovus Stylus Common UI Text Command Reference*
- RC Extraction chapter in the *Innovus Stylus Common UI User Guide*

set_extract_shrink_factor

```
set_extract_shrink_factor
[-help]
shrink_factor
```

Sets the value by which the software shrinks the design before extraction. The shrink factor is used only for extraction calculations. It does not change the actual design.

Note: You do not need to specify the shrink factor if the capacitance table file in use contains the shrink factor specification in its header.

Parameters

`-help`	Outputs a brief description that includes type and default information for the `set_extract_shrink_factor` parameter. For a detailed description of the command and its parameter, use the man command: `man set_extract_shrink_factor`.
`shrink_factor`	Specifies the value by which the software shrinks a design. *Default:* `1.0` *Data_type*: float, required

Example

The following command sets the shrink factor to 0.9:

```
set_extract_shrink_factor 0.9
```

Setting the shrink factor to 0.9 reduces the wire lengths, widths, and spacings in the design by 0.9 times the original values.

Related Information

- `extract_rc`
- `design Category Attributes`
- extract_rc Category Attributes chapter of *Innovus Stylus Common UI Text Command Reference*
- RC Extraction chapter of *Innovus Stylus Common UI User Guide*

write_cap_table

```
write_cap_table
[-help]
[-cap total_cap_factor]
[-encrypt]
-out_cap_table_file file_name
[-res resistance_factor]
[-xcap cross_coupling_factor]
[-lef file_name [-shrink_factor value]]
{-ict file_name | -in_cap_table file_name}
```

Reads in the interconnect technology (ICT) process file and generates a Capacitance table file. The capacitance table file contains a header having a summary of process information and manufacturing effects followed by tables containing coefficients used during RC extraction.

The capacitance table is used both for preRoute extraction (above 32nm) and postRoute extraction (`extract_rc_effort_level low`).

Note: For TQuantus, IQuantus, preRoute extraction (32nm and below), and Standalone Quantus, a Quantus Techfile is needed instead of a capacitance table.

The `write_cap_table` command automatically omits the creation of coefficients for the poly layers, local interconnect layers, and mimcap layers. The `write_cap_table` command recognizes poly layers by the `gate_forming_layer true` statement.

Note: The command accepts ICT files that contain the `gate_forming_layer true` construct in multiple `conductor` sections to allow handling of diffusion models as a conductor. In this case, the Innovus captable generation process ignores all but the last `gate_forming_layer true` construct. The last one is considered as `poly` and is used as the ground plate level for the metal layers above.

ⓘ The command does not support `grown_from_conformal` (non-planar stack) construct defined in the ICT file.

ⓘ You can use the multi-CPU processing commands to generate a capacitance table in the parallel mode. However, this functionality is only available when invoking `write_cap_table` from the software prompt, and not from the standalone utility `write_cap_table`. To generate a capacitance table file in the parallel mode, you must first set up the multiple-CPU processing mode using the multiple-CPU processing commands.

See the following multiple-CPU processing commands for more information:

- `set_distributed_hosts`
- `set_multi_cpu_usage`

For more information, see the "Distributed Processing" section of the "RC Extraction" chapter in the *Innovus Stylus Common UI User Guide*.

Examples

- To run the `write_cap_table` command in the parallel mode on different hosts, specify the following commands:

  ```
  set_distribute_host -rsh -add { host1 host2 host3 }
  set_multi_cpu_usage -remote_host 3
  write_cap_table -ict sample.ict -out_cap_table_file sample.capTbl
  ```

- To run the `write_cap_table` command with three threads, specify the following commands:

  ```
  set_distribute_host -local
  set_multi_cpu_usage -local_cpu 3
  write_cap_table -ict sample.ict -out_cap_table_file sample.capTbl
  ```

Parameters

`-help`	Outputs a brief description that includes type and default information for each `write_cap_table` parameter. For a detailed description of the command and all of its parameters, use the man command: `man write_cap_table`.
`-cap` *`total_cap_factor`*	Specifies the capacitance scale factor. Use this parameter with the -in_cap_table parameter to generate a capacitance table with capacitance values scaled by the specified scale factor. *Data_type*: float, optional
`-encrypt`	Encrypts the generated capacitance table to protect the process information in the capacitance table header. When you use the `create_rc_corner -cap_table` command to read the encrypted capacitance table, the software writes the version number of the software that you had used to generate the file, and the name of the process to the log file. *Data_type*: bool, optional
`-ict` *`file_name`*	Specifies the name and path to the ICT (interconnect technology) process file. To support multiple process corners, a capacitance table should be created for each corner from the matching corner ICT file. *Data_type*: string, optional
`-in_cap_table` *`file_name`*	Specifies the name of an existing capacitance table in ASCII format. You use this parameter along with the -encrypt parameter to create an encrypted version of an existing capacitance file. *Data_type*: string, optional
`-lef` *`file_name`*	Specifies the name and the full path to the process LEF file for optimal accuracy. With the LEF file, this parameter generates a capacitance table by using the actual wire widths as specified in the non-default rules defined in the LEF. Without the LEF, the software uses generic widths and spacings, which might increase the need for interpolation during extraction. *Data_type*: string, optional
`-out_cap_table_file` *`file_name`*	Specifies the output filename that contains the capacitance table. *Data_type*: string, optional
`-res` *`resistance_factor`*	Specifies the resistance scale factor. Use this parameter with the -in_cap_table parameter to generate a capacitance table with resistance values scaled by the specified scale factor. *Data_type*: float, optional
`-shrink_factor` *`value`*	Specifies the value by which the measurements in the optional LEF file have to be shrunk for the technology used with this design. Use this parameter only with the `-lef` parameter and when the LEF technology does not match the target technology in the ICT file. Innovus scales the dimensions when reading the LEF file using the shrinkFactor specified. The ICT file should always refer to the target technology (min width and min spacing). **Note**: When working on a design that needs to be scaled down in Innovus, you must also set the shrinkfactor in *Design Import*. This tells the extractor to scale down all geometries before processing. When the ICT file contains a layout_scale in the process declaration part, you do not need to specify the shrink factor. In this case, the embedded shrink factor will also be written in the captable header. Also, there is then no need to set the shrink factor in the Design Import form. The embedded layout_scale is used to support half nodes such as 40nm technology. *Default:* 1.0 *Data_type*: float, optional
`-xcap` *`cross_coupling_factor`*	Specifies the cross-coupling capacitance scale factor. Use this parameter with the -in_cap_table parameter to generate a capacitance table with cross-coupling capacitance values scaled by the specified scale factor. *Data_type*: float, optional

Example

The following command reads in the ICT file `test.ict`, and generates a `new.capTbl` output file:

```
write_cap_table -ict test.ict -out_cap_table_file new.CapTbl
```

Related Information

- `create_rc_corner`
- `extract_rc`
- `extract_rc Category Attributes`
- `set_distributed_hosts`
- `set_multi_cpu_usage`
- The following section in RC Extraction chapter of *Innovus Stylus Common UI User Guide*
  - Generating a Capacitance Table

write_extraction_spec

```
  write_extraction_spec
[-help]
[-no_design_data]
[-out_dir {dir_name}]
[-rc_corners {list_of_rc_corners}]
```

Creates the Quantus command file or the Common Command Language (CCL) file in the current working directory and saves the design data for running standalone Quantus. The command may be specified with a directory path where technology library file, LEF list file (for DEF-based flow), and DEF or OA design will be stored. However, if no directory path is specified, then a directory will be created in the current directory. The name of the directory will be in the format: `tmp_qrc_pid_time`. You can also specify whether you want to save the design data or not by using the `-no_design_data` parameter. If it is specified, then no design data, such as DEF or OA design will be saved.

In addition, to generate a command file for the OA-based flow, ensure that the `extract_rc_use_qrc_oa_interface` attribute is set to `true`.

Note: Run this command after loading the design, setting up the MMMC environment, and specifying the CCL layer map file, if required.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `write_extraction_spec` parameter. For a detailed description of the command and all of its parameters, use the man command: `man write_extraction_spec`.
`-no_design_data`	When this parameter is specified, no design data, such as DEF or OA design will be saved. By default, the design data will be saved. If this option is specified, then follow the steps listed below to run Quantus. 1. For DEF-based flow, provide the DEF file while invoking standalone Quantus. 2. For OA-based flow, the command file that is generated will assume an OA design in the following directory: `<lib name>/<top level cell name>/for_qrc`. The value of `<lib name>` and `<top level cell name>` will be written into the command file that is generated. If you want to specify a different OA design path, modify the command file. **Note**: It is recommended that you use this parameter if you already have the DEF or OA design saved. *Data_type*: bool, optional
`-out_dir {dir_name}`	Specifies the location of the output directory in which the technology library file and associated data, and design data will be saved. If no path is specified, then a directory, `tmp_qrc_pid_time` will be created in the current directory by default. *Data_type*: string, optional
`-rc_corners {list_of_rc_corners}`	Specifies the list of RC corners for which the Quantus command file is created. By default, the command outputs all active RC corners. When this parameter is specified, only the specified corners are included in the Quantus command file. For example, Consider two RC corners, `RC1` and `RC2`, of which only `RC1` is active. • If this parameter is not specified, then the command will generate a CCL file with one active corner, `RC1`. • If this parameter is specified with a non-active corner, `RC2`, then the command will create a CCL file for the specified corner. `write_extraction_spec -rc_corners RC2` *Default*: `all active corners` *Data_type*: (rc_corner)+, optional

Related Information

- `extract_rc`
- extract_rc Category Attributes chapter of *Innovus Stylus Common UI Text Command Reference*
- The following section in RC Extraction chapter of *Innovus Stylus Common UI User Guide*
  - Sign-Off Extraction Using Quantus

write_parasitics

```
write_parasitics
[-help]
[-exclude_ilm]
[-no_resistances]
[-use_eeq_cell_with_liberty_info]
{{{-spef_file file_name} [-exclude_nets_file file_name | -include_nets_file file_name | -net_name net_name]}}
[-view view_name | -rc_corner rc_corner_name]
[-cap_unit {pf | ff}]
```

Reads the parasitic database and outputs the parasitics in the Standard Parasitic Exchange Format (SPEF).

Parameters

`-help`	Outputs a brief description that includes type and default information for each `write_parasitics` parameter. For a detailed description of the command and all of its parameters, use the man command: `man write_parasitics`.
`-cap_unit {pf | ff}`	Specifies the unit of capacitance as `fF` or `pF` in the output SPEF file. *Default:* `pF` *Data_type*: enum, optional

`-exclude_ilm`	Writes out only the top-level SPEF while excluding the ILM parasitics.
	Note: When this parameter is specified, the top cell-name is suffixed in the SPEF output filename.
	For example, if `test.spef` is the name of the SPEF file and `top_cell_name` is the cell-name, then when you run below command:
	`write_parasitics -spef test.spef -exclude_ilm`
	The output SPEF filename will be:
	`test.spef.top_cell_name`

`-exclude_nets_file` *file_name*	Specifies the name of the file that contains nets to be excluded from the output file. *Data_type*: string, optional
`-include_nets_file` *file_name*	Specifies the name of the input file that contains the net names to be extracted. *Default:* Extracts all net names *Data_type*: string, optional
`-net_name` *net_name*	Specifies the name of the net to be written in the SPEF file. *Default:* Writes all nets *Data_type*: string, optional
`-no_resistances`	Generates a SPEF file without the resistance information. The SPEF file generated by this parameter can only be used to improve the performance of third-party simulator tools in early stages of the design. **Caution**: Most tools expect the resistance information to be present in the SPEF file for performing extraction. Therefore, SPEF files with no resistance information should not be used with Ostrich for RC correlation. *Data_type*: bool, optional
`-rc_corner` *rc_corner_name*	Specifies the RC corner that is used to generate the parasitic output file in the multi-corner flow. *Data_type*: string, optional
`-spef_file` *file_name*	Specifies the name of the SPEF output file. If you add `.gz` as a suffix, the `write_parasitics` command automatically compresses the generated file. *Data_type*: string, optional
`-use_eeq_cell_with_liberty_info`	Writes EEQ master cell in case the EEQ cells are present. *Data_type*: bool, optional
`-view` *view_name*	Specifies that the RC corner associated with the specified view is used to generate the output SPEF file in the multi-corner flow. *Data_type*: string, optional

Examples

- Reads the parasitic database created during RC extraction and writes the interconnect capacitance in the SPEF format to the file `TOPCHIP_SP.spef` for RC corner, `corner1`:

  ```
  write_parasitics -rc_corner corner1 -spef_file TOPCHIP_SP.spef
  ```

Related Topics

- `read_spef`
- The following sections in RC Extraction chapter of *Innovus Stylus Common UI User Guide*
  - Extraction Flow in Innovus
  - Correlating Native Extraction With Sign-Off Extraction

write_rcdb

```
 write_rcdb
[-help]
out_dir
```

Saves the RC extraction data in the specified directory. Use the `read_rc` command to restore the parasitic data. Use this command after running either `extract_rc` or `read_spef` to ensure that the parasitic data is available in the Innovus database.

Note: You can restore the parasitic data on systems that are same as those used for saving the data. For example, if the data was saved on Linux in 32-bit mode, you can restore it on Linux 32-bit mode only.

Parameters

`-help`	Outputs a brief description that includes type and default information for the `write_rcdb` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man write_rcdb`.
`out_dir`	Specifies the location of the RCDB directory. The directory name should follow the `*.rcdb.d` format. *Data_type*: string, optional

Related Information

- `read_rc`
- `report_rcdb`
- RC Extraction chapter of *Innovus Stylus Common UI User Guide*

33

Route Commands and Attributes

- add_tracks
- check_tquantus_model_file
- check_tracks
- colorize_cell_library
- create_channel_density_control
- create_tquantus_model_file
- delete_shields
- fix_drc_with_patch
- get_route_attributes
- report_congested_area
- report_route
- report_shields
- report_tracks
- report_wires
- route_add_shields
- route_design
- route_detail
- route_early_global
- route_early_global Category Attributes
- route_eco
- route_fix_signoff_drc
- route_global_detail
- route Category Attributes
- set_max_route_layer
- set_route_attributes
- write_congested_area_report
- write_litho_hotspots
- write_nets_in_congested_area

add_tracks

```
add_tracks
[-help]
[-honor_pitch]
[-keep_default_width]
[-mask_pattern {{layer1  mask [mask …]} [layer2 ...] }]
[-mode {replace append}]
[-offsets {layer1 horiz|vert [die_box] offset} [{layer2 …}] ...}]
[-pitch_pattern {layer1 offset offset pitch pitch [pitch pitch …]} [{layer2 …}] ...}]
[-pitches {layer1 horiz|vert pitch} [{layer2 horiz|vert pitch}] ...}]
[-rect_only_pattern {{layer1 [pattern ...]} {layer2 [pattern ...]} .. }]
[-route_rule route_rule_name]
[-snap_m1_track_to_cell_pins]
[-width_pitch_pattern {{layer1 offset offset width width pitch pitch [width width pitch pitch]...} [{layer2 …}] ...}]
```

Deletes existing routing tracks and generates new routing tracks based on the technology data or user specific values. If no parameters are given to the command, it examines the LEF layer offset and pitch values, but may adjust them based on the LEF width and spacing rules, via-rules, or the width and height of the standard cells to create tracks optimized for routing. If you want to override the automatically generated pitch and offset values, you should use the parameters below:

- Use `-offsets` and `-pitches` to create uniform track patterns.
- Use `-track_pattern` or `-track_width_pattern` for non-uniform track patterns.

The Innovus software generates tracks automatically. You do not have to run this command unless you want to override the defaults or you import a DEF file using the `read_def` command and you want to overwrite the track definitions in the DEF file. To overwrite the track definitions, run this command after running the `read_def` command. Various warning messages are generated if any values given will likely give poor routing results (For example, A pitch less than minimum wire-to-via spacing, or a bottom routing layer tracks that do not align to the standard-cell heights.)

- Non-preferred routing tracks of each layer are aligned with the preferred routing tracks of the next lower layer.
- Non-preferred routing tracks of the first layer are aligned with the tracks of the next upper layer.

If the LEF FIRSTLASTPITCH property (or OA equivalent) is defined for a layer, then non-uniform tracks will be created with the first and last pitch of the tracks set to align to the standard cell rows. This makes the tracks aligned with the standard-cell power rails have a wider-pitch on both sides to allow a wider power-route with min-spacing to the next routing track.

Parameters

`-help`	Prints a brief description that includes the type and default information for each `add_tracks` parameter. For a detailed description of the command and all of its parameters, use the man command: `man add_tracks`.
`-honor_pitch`	Honors the pitch specified in the LEF file. By default, the `add_tracks` command adjusts the pitch to what it thinks is best. The `-honor_pitch` parameter forces `add_tracks` to use the layer pitch value specified in the LEF or OA technology file. If the layer pitch specified in the LEF file is smaller than the minimum line-to-via pitch, the pitch is reset to the minimum line-to-via pitch and a warning message is displayed in the log file. **Note:** The `-honor_pitch` parameter cannot be used with any of the options setting pitch values directly (`-pitches, -pitch_pattern, -width_pitch_pattern`). *Data_type*: bool, optional
`-keep_default_width`	Keeps user specified default width between routing tracks. **Note:** This enables the interoperability of width spacing patterns with Virtuoso. *Data_type*: bool, optional
`-mask_pattern {{layer1  mask [mask …]} [layer2 ...] }`	

	Specifies the mask pattern for the specified layers for layers that have multiple masks. The mask values are applied to the preferred direction tracks, and then repeat. There is no color for non-preferred direction tracks. If `-pitch_pattern`, or `-width_pitch_pattern` is given, the number of elements in `-mask_pattern` must match for each layer or an error occurs. • *`layer`*: You can use LEF layer names or built-in routing-layer names (Metal1 for the first routing layer, Metal2 for the second layer) for the *`layer`* name. • *`mask:`* Specify 1, 2, or 3. The layer must be defined to use the specified number of masks. For example, `-mask_pattern {metal1 1 2 3 1 2 metal2 1 2 1 2 1}` *Data_type*: string, optional
`-mode {replace append}`	Enables incremental replace/append operations for tracks and regenerates tracks for specified layers only. In the incremental mode, you can replace or append any specific tracks one-by-one. You do not need to specify a long list of track definitions anymore. • `replace`: Enables the `add_tracks` command to remove all the tracks in the specified layers and then only generate/add specified tracks from the command line. • `append`: When specified, the `add_tracks` command does not remove any tracks. It will append specified tracks into existing tracks. *Data_type*: enum, optional
`-offsets {`*`layer1`* `horiz|vert [die_box]` *`offset`*`} [{`*`layer2`* `…}] ...}`	
	Specifies the horizontal offset or vertical offset for the specified layers. • *`layer`*: You can use LEF layer names or built-in routing-layer names (Metal1 for the first routing layer, Metal2 for the second layer) for the *`layer`* name. • *`offset`*: Specifies the offsets (in microns) for the specified layer. **Note:** The value of *`offset`* is relative to the lower-left corner of the core_box that is aligned to the standard cell rows, and can be a negative number. If die_box is given, the *`offset`* is relative to the lower-left corner of the die_box rather than the core_box. **Note:** die_box should not be used for the first two routing layers that need to align with the standard cells pins, but it can be used to align some of the upper routing layer tracks with a power-route grid structure. For example, `-offsets {metal1 horiz 0.1 metal2 vert 0.0 metal4 vert die_box 0.3}` *Data_type*: string, optional
`-pitch_pattern {`*`layer1`* `offset` *`offset`* `pitch` *`pitch`* `[pitch` *`pitch`* `…]} [{`*`layer2`* `…}] ...}`	
	Specifies an offset, and pitch pattern for the tracks in the preferred routing direction when non-uniform spacing is needed. These values override any `-pitches` or `-offsets` values for this layer in the preferred direction. • *`layer`*: You can use LEF layer names or built-in routing-layer names (Metal1 for the first routing layer, Metal2 for the second layer) for the *`layer`* name. • *`offset`*: Specifies the offsets (in microns) for the specified layer. • *`pitch`*: Specifies the pitch value. You can repeat the last pitch value using `repeat` *`num`*. It is best to add {} to make it clear which pitch value is repeated. **Note**: {} are ignored but allowed to make it easier to read the command. For example, `-pitch_pattern {metal1 offset 0.2 pitch 0.2 {pitch 0.1 repeat 9} pitch 0.2 metal2 offset 0.1 pitch 0.2 {pitch 0.1 repeat 8}` } *Data_type*: string, optional
`-pitches {`*`layer1`* `horiz|vert` *`pitch`*`} [{`*`layer2`* `horiz|vert` *`pitch`*`}] ...}`	

	Specifies the horizontal pitch or vertical pitch for the specified layers. This is only used for layers that have uniform track spacing. • *layer*: You can use LEF layer names or built-in routing-layer names (Metal1 for the first routing layer, Metal2 for the second layer) for the *layer* name. • *pitch*: Specifies the pitch value (in microns). **Note**: {} are ignored but allowed to make it easier to read the command. For example, `-pitches {metal1 vert 0.1 metal2 horiz 0.1 metal4 horiz 0.3}` *Data_type*: string, optional
`-rect_only_pattern {{layer1 [pattern ...]} {layer2 [pattern ...]} .. }`	

	Enables you to specify which subset of tracks would only allow rectangular shapes (rect_only).
	Note: In the GUI, 'rect_only' is added behind the width to indicate that the track has the “RECTONLY PARTIALTRACKS” LEF property assigned to it. The presence of the *rect_only* tag indicates that the track has the 'rectangular shape only' restriction and should be honored in routing.
	• *layer*: The layer must be defined to use the rect_only routing pattern.
	• *pattern*: Use 0 or 1 to specify the rect_only routing pattern.
	  ◦ 0: Indicates no rect_only in generated tracks.
	  ◦ 1: Indicates rect_only in generated tracks.
	For example,
	`add_tracks -rect_only_pattern {m2 0 1 1 1 0 0}`
	
	Data_type: string, optional

`-route_rule route_rule_name`	Specifies the name of a non-default route rule. When specifies, the existing tracks are not deleted.Instead, an additional set of tracks are created for use by this *route_rule_name* on top of the existing tracks.
	Note: When the `-route_rule` parameter is used, you must specify everything for this specific *route_rule_name* in one command, irrespective of what `-mode` option you specify.
	Example:
	In the following command, M6 and M8 NDR tracks are added in one command. These should ***NOT*** be done in the two commands.
	`set addTrack_pitch_pattern_2w "{\`
	`{m6 offset 0.490 width 0.076 pitch 0.076 width 0.076 pitch 0.076 width 0.076 pitch 0.076 width 0.076 pitch 0.076 width 0.076 pitch 0.304} \`
	`{m8 offset 0.490 width 0.076 pitch 0.076 width 0.076 pitch 0.076 width 0.076 pitch 0.076 width 0.076 pitch 0.076 width 0.076 pitch 0.304}}"`
	`add_track -width_pitch_pattern $addTrack_pitch_pattern_2w -mode append -route_rule {TrunkNDR_2W}`
	Note: To later modify/append NDR tracks, you must delete all tracks (including default and NDR tracks) using the `delete_tracks` command and then regenerate all of them.
	Data_type: string, optional

`-snap_m1_track_to_cell_pins`	Extracts TSMC N10 pins to generate tracks in Metal1 layer. When specified, it scans all standard cells in the library and tries to extract all possible rectangle pins shapes and mask numbers. Tracks are created to hit the center of these pins. All tracks are assigned a mask based on the pin mask value, if specified. *Data_type*: bool, optional
`-width_pitch_pattern { {layer1 offset offset width width pitch pitch [width width pitch pitch]...} [{layer2 …}] ...}`	

	Specifies track width patterns for the preferred direction. Use { } to specify track width pattern for more than one layer. Specifies the width and pitch pattern for the tracks in the preferred routing direction, for technologies that need a width value attached to each track. These values override any `-pitches` or `-offsets` values for this layer in the preferred direction. If `-pitch_pattern` is given for any of the layers, `-width_pitch_pattern` will override the `-pitch_pattern` values. • *`layer`*: You can use LEF layer names or built-in routing-layer names (Metal1 for the first routing layer, Metal2 for the second layer) for the *`layer`* name. • *`width`*: Specifies the width value. • *`pitch`*: Specifies the pitch value (in microns). **Note:** You can repeat the last width and pitch value using `repeat num`. It is best to add {} to make it clear which value are repeated. For example, `-width_pitch_pattern {metal1 offset 0.2 width 0.2 pitch 0.2 {width 0.1 pitch 0.1 repeat 8} width 0.1 pitch 0.2 metal2 offset 0.1 width 0.2 pitch 0.2 {width 0.1 pitch 0.1 repeat 8} width 0.1 pitch 0.2 }` *Data_type*: string, optional

Examples

- The following command deletes any existing tracks, and generates new routing tracks based on the LEF or OA layer offset and pitch settings, and the standard-cell pins.
  ```
  add_tracks
  ```
- The following command generates uniform vertical tracks on metal1 with a horizontal offset of 0.1 from the core_box, a horizontal pitch of 0.2, and horizontal tracks on metal 2 with a vertical offset of 0.2 from the core_box, and a vertical pitch of 0.3. Other layers will use the LEF or OA layer settings.
  ```
  add_tracks -offsets {metal1 horiz 0.1 metal2 vert 0.2} -pitches {metal1 horiz 0.2 metal2 vert 0.3}
  ```
- The following command generates a non-uniform pattern of tracks for the preferred routing direction, using the values given. Extra {} are added to make it more readable.
  ```
  add_tracks -pitch_pattern {metal1 offset 0.1 pitch 0.2 {pitch 0.1 repeat 8} metal2 offset 0.2 pitch 0.2 {pitch 0.1 repeat 8} }
  ```
- The following command generates a non-uniform pattern of tracks with width values for the preferred routing direction, using the values given settings.
  ```
  add_tracks -width_pitch_pattern \
  {{metal1 offset 0.1 width 0.2 pitch 0.2 {width 0.1 pitch 0.1 repeat 8} width 0.1 pitch 0.2} \
  {metal2 offset 0.2 width pitch 0.2 {pitch 0.1 repeat 8}} }
  ```

Pitch Rule Examples

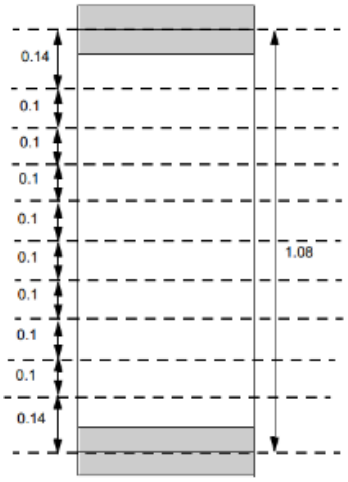
Example 1

The following example shows how routing_tracks are created for a layer with FIRSTLASTPITCH.
Suppose the LEF has this statement for the metal1 layer.

```
PROPERTY LEF58_PITCH STRING "
    PITCH 0.1 FIRSTLASTPITCH 0.14 ; ";
```

Then the horizontal metal1 tracks will be created as shown below by default, with no extra parameters passed to add_tracks.

Example 2



In this example, the cell height is 1.08. The dotted line represents the routing tracks. The first and last tracks are 0.14 from the cell row boundary while the intermediate tracks are 0.1 apart making 2 * 0.14 + 8 * 0.1 = 1.08. The routing track definition would be repeated based on the cell rows.

Related Information

- `report_tracks`
- `check_tracks`
- Using the NanoRoute Router chapter in the *Innovus Stylus Common UI User Guide*
  - Adding Tracks
  - Running the NanoRoute Router with Innovus Text Commands

check_tquantus_model_file

```
check_tquantus_model_file
[-help]
```

Checks the TQuantus model file created using the `create_tquantus_model_file` command. It ensures that the TQuantus model file is consistent with the following input files inside the Innovus database:

- QRC techfiles
- Pitch information from the technology LEF file
- RC corners

The following commands invoke this command automatically to check the consistency of the TQuantus model file if TQuantus is chosen as the extraction engine.

- `route_design -trackOpt`
- `opt_design -postRoute`
- `time_design`
- `extract_rc`

Note: The software issues warnings if inconsistencies are detected between the TQuantus file and the RC corner information, QRC techfile, or pitch information in the LEF technology file. If such warnings are issued, the TQuantus model file will have to be re-generated manually.

Parameter

`-help`	Outputs a brief description of the `check_tquantus_model_file` parameter. For a detailed description of the command, use the man command `man check_tquantus_model_file`.

Related Information

- `create_tquantus_model_file`
- `route_design`
- `opt_design`
- `time_design`
- `extract_rc`
- Using the NanoRoute Router chapter in the *Innovus Stylus Common UI User Guide*
  - Creating RC Model Data in TQuantus Model File

check_tracks

```
check_tracks
[-help]
[-direction {horiz | vert | H | V}]
[-grid spacing]
[-layer layer_name]
[-mask mask_name]
[-offset string]
[-origin {design | core | D | C}]
```

Checks a predefined set of tracks to verify their routing track offset, direction, and grid. It reports the following conditions in the Innovus log file:

1. Tracks whose `STEP` is smaller than the minimum pitch for the designated layer.
2. The percentage of on-track pins and pins that are not accessible (for regular routing).
3. Fully blocked pins and potential off-grid track pins (for special routing).

Before running this command, complete the following steps:

- Load a LEF file with the library and technology information.
- Place the design.

Parameters

`-direction {horiz | vert | H | V}`	Specifies the direction of the tracks to check. *Data_type*: string, optional
`-grid spacing`	Specifies the track spacing. *Data_type*: string, optional
`-help`	Outputs a brief description that includes type and default information for each `check_tracks` parameter. For a detailed description of the command and all of its parameters, use the man command: `man check_tracks`
`-layer layer_name`	Specifies the track layer. *Data_type*: string, optional
`-mask mask_name`	Specifies the mask name. *Data_type*: string, optional
`-offset string`	Specifies the track offset from the starting point. *Data_type*: string, optional
`-origin {design | core | D | C}`	Specifies the track starting point. *Data_type*: string, optional

Examples

- The following command checks the tracks in the horizontal direction on layer M3 with a grid spacing of 3.
  ```
  check_tracks -direction H -layer M3 -grid 3
  ```

  The following output displays on screen:

  ```
  Total 35 tracks meet specification: layer=Metal3 dir=horiz offset=0 grid=3
  Total 2622 tracks not hit under specified command line options
  ```

Related Information

- `add_tracks`
- `report_tracks`
- Using the NanoRoute Router chapter in the *Innovus Stylus Common UI User Guide*
  - Resolving Open Nets
  - Diagnosing Problems Using check_tracks

colorize_cell_library

```
colorize_cell_library
[-help]
[-cell cell_name]
[ [-reset ] | [-enforce ] ]
```

Colors each standard cell in the library, using the pre-color method.

Parameters

`-cell` *cell_name*	Specifies the name of the cell to colorize. The `colorize_cell_library` command only targets the specified cell, irrespective of the cell type. *Data_type*: (libCell/ptnCell)+, optional
`-help`	Outputs a brief description that includes the type and default information for each `colorize_cell_library` parameter. For a detailed description of the command and all of its parameters, use the man command `man colorize_cell_library`.
`-enforce`	Specifies that re-colorization will be enforced. It colors the cell even if it is already colored. *Data_type*: string, optional
`-reset`	Resets the library cell color data. It cleans the color of every standard cell, except the LEF-color. *Data_type*: bool, optional

Example

- The following command resets the library cell color data.
  ```
  colorize_cell_library -reset
  ```

Related Information

- `route` Category Attributes
- Using the NanoRoute Router chapter in the *Innovus Stylus Common UI User Guide*

create_channel_density_control

```
create_channel_density_control
[-help]
{lower_left_x} {lower_left_y} {upper_right_x} {upper_right_y} layer ratio
```

Controls the routing density for a specified area on a layer by setting a maximum density percentage that Early Global Route cannot exceed. Use the `create_channel_density_control` command after performing placement.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `create_channel_density_control` parameter. For a detailed description of the command and all of its parameters, use the man command `man create_channel_density_control`.
layer	Specifies the layer on which to specify the routing density control. **Note:** Layers can be specified using the layer names or layer ID numbers. *Data_type*: string, required
{*lower_left_x*}	Specifies the lower left x coordinate of the density area. *Data_type*: float, required
{*lower_left_y*}	Specifies the lower left y coordinate of the density area. *Data_type*: float, required
ratio	Specifies the maximum routing density allowed for the area, as a percentage. *Min*: 0 *Max:* 100 *Value:* `0` (zero percent) to `1` (100 percent) *Data_type*: float, required
{*upper_right_x*}	Specifies the upper right x coordinate of the density area. *Data_type*: float, required
{*upper_right_y*}	Specifies the upper right y coordinate of the density area. *Data_type*: float, required

Examples

- The following command creates a routing density control area on layer M3 with a maximum density of 50 percent:
  ```
  create_channel_density_control 10 10 500 30 M3 0.50
  ```

Related Information

- `route_early_global`
- `route_early_global` Category Attributes
- Using Early Global Route for Congestion and Timing Analysis chapter in the *Innovus Stylus Common UI User Guide*

create_tquantus_model_file

```
create_tquantus_model_file
[-help]
-out_file rc_model_file_name
[-reset]
```

Creates the RC model data and stores it in a TQuantus model file. This data is used by the RC extractor. This command eliminates the need for the RC extractor to rebuild the RC models every time it runs. It can just use the model data created using this command. Before creating the TQuantus model file, ensure the following:

- QRC techfiles are available
- Pitch information from the technology LEF file is available
- All RC corners are set up

The consistency of the created TQuantus model file is checked using the `check_tquantus_model_file` command.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `create_tquantus_model_file` parameter. For a detailed description of the command and all of its parameters, use the man command `man create_tquantus_model_file`.
`-out_file` *rc_model_file_name*	
	Writes the created RC model data into a file with the specified file name. *Data_type*: string, optional
`-reset`	Resets the RC model file. *Data_type*: bool, optional

Related Information

- `check_tquantus_model_file`
- Using the NanoRoute Router chapter in the *Innovus Stylus Common UI User Guide*
  - Creating RC Model Data in TQuantus Model File

delete_shields

```
delete_shields
[-help]
{-nets list_of_signal_nets | -selected}
[-routes_only {0 | 1}]
```

Deletes physical shielding wires from the specified or selected signal nets and removes `SHIELDNET` attributes from the nets. Use the `delete_shields` command after setting the `SHIELDNET` attribute on a net. To set the `SHIELDNET` attribute, use the `set_route_attributes` command.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `delete_shields` parameter. For a detailed description of the command and all of its parameters, use the man command `man delete_shields`.
`-nets list_of_signal_nets`	Specifies a list of signal nets with one or more `SHIELDNET` attributes. You can use the asterisk (`*`) wildcard in the net names. *Data_type*: string, optional
`-routes_only {0 | 1}`	Specifies that only shield wires and vias will be deleted. *Data_type*: enum, optional
`-selected`	Delete shield of selected nets. *Data_type*: bool, optional

Examples

- The following command deletes `SHIELDNET` attributes and physical shielding wires from net `clk1`:
  ```
  delete_shields -nets clk1
  ```
- The following command deletes `SHIELDNET` attributes and physical shielding wires from nets `c*`:
  ```
  delete_shields c*
  ```
- The following command deletes `SHIELDNET` attributes and physical shielding wires from selected nets:
  ```
  delete_shields -selected
  ```

Related Information

- `set_route_attributes`
- `route_add_shields`
- `report_shields`
- Using the NanoRoute Router chapter in the *Innovus Stylus Common UI User Guide*

fix_drc_with_patch

```
fix_drc_with_patch
[-help]
[area]
[-layer_range bottomLayer:topLayer]
```

Fixes violations by adding trim/patch in the whole design which includes fixed cell objects, preRouted nets, and special nets on a preRoute db. You can use this command after floorplanning the design or after PG routing or on preRouted routing database, however, you must use it before running `route_design`.

Note: Cadence recommends you to use the `fix_drc_with_patch` command to check the sign-off on preRoute db. It should not be used for flow continuation.

Parameters

area	Enables the tool to trim/add patches and fix DRC violations in the specified route area. You can specify the area using the *x1 y1 x2 y2* coordinate values. *Data_type*: box, optional
-help	Outputs a brief description that includes the type and default information for each `fix_drc_with_patch` parameter. For a detailed description of the command and all of its parameters, use the man command: `man fix_drc_with_patch`
-layer_range *bottomLayer:topLayer*	Enables the tool to trim/add patches and fix DRC violations within the specified layer range (*bottomLayer*:*topLayer*). *Type:* string, optional

Examples

The following command adds patches and fixes DRC violations within the 2:4 layer range:

```
fix_drc_with_patch -layer_range 2:4
```

Related Topic

- `route_eco`
- `route_design`
- Using the NanoRoute Router chapter in the *Innovus Stylus Common UI User Guide*

get_route_attributes

```
get_route_attributes
[-help]
-nets net_name
[-quiet]
```

Displays the current net attribute settings in the Innovus console and in the Innovus log file. At the end of the display, the NanoRoute router lists each attribute, enclosing it in braces ({}) so you can use it in a Tcl script with variables. See `set_route_attributes` for descriptions of NanoRoute attributes.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `get_route_attributes` parameter. For a detailed description of the command and all of its parameters, use the man command `man get_route_attributes`.
`-nets net_name`	Specifies the net for which to display attributes. *Data_type*: string, required
`-quiet`	Displays the current settings for the specified parameters in Tcl list format only. If you specify `-quiet` without any parameters, the software displays the current settings of all `set_route_attributes` parameters in Tcl list format. *Data_type*: bool, optional

Example

The following command displays the attributes for `net1`:

```
get_route_attributes -nets net1
```

The Innovus software returns the following:

```
#NET net1 attributes:
# weight = 2 (default)
# non_default_rule = default (default)
# shield_net = none (default)
# pattern = steiner (default)
# top_preferred_routing_layer = 15 (default)
# bottom_preferred_routing_layer = 0 (default)
# preferred_extra_space = 0 (default)
# avoid_detour = false (default)
# skip_routing = false (default)
# skip_antenna_fix = false (default)
# si_post_route_fix = false (default)
{weight 2} {non_default_rule default} {shield_net none} {pattern steiner} {top_preferred_routing_layer 15} {bottom_preferred_routing_layer
0} {preferred_extra_space 0} {avoid_detour false} {skip_routing false} {skip_antenna_fix false} {si_post_route_fix false}
```

Related Information

- `set_route_attributes`
- Using the NanoRoute Router chapter in the *Innovus Stylus Common UI User Guide*

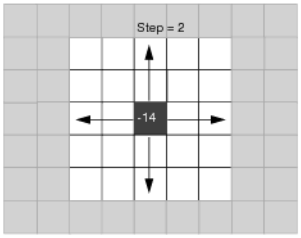
report_congested_area

```
report_congested_area
[-help]
[-num [all | numOfHotSpot]]
[-cutoff_ratio cutOffRatioOfPeakHotSpot]
[-cutoff_value cutOffCongestNum]
[-mode modeName]
[-step num]
[-cutoff_area cutOffGcellNum]
[-out_file fileName]
```

Reports the congestion hot spots in the design.

The Innovus software uses the following algorithm to define the boundaries of hot spots:

1. Finds the most over-congested gcell (that is, the gcell with the largest overflow value) in the design. A gcell has overflow if its demand exceeds its supply. Typically, the supply is the number of unobstructed tracks crossing the gcell, and the demand is the number of wires assigned to it.
2. Starting from the over-congested gcell, checks a specified number of gcells (using `-step`) in one direction for congestion. If the gcells have an overflow value that is less than the specified congestion threshold (using `-cutoff_ratio` or `-cutoff_value`), the software determines this to be the boundary for that side of the hot spot, and begins checking in a new direction.



3. After the software establishes the boundaries for the first hot spot, it finds the next most over-congested gcell that is not part of the already defined hot spot and begins defining the boundaries for it. You can use the `report_congested_area` command after running Early Global Route.

Parameters

`-cutoff_area` *`cutOffGcellNum`*	
	Filters out any hot spots that have an area (specified in gcells) that is less than *`cutOffGcellNum`*. *Default:* `0` (Reports all hot spots in the design) *Data_type*: int, optional
`-cutoff_ratio` *`cutOffRatioOfPeakHotSpot`*	
	Specifies the minimum congestion threshold value a gcell must have to be considered part of the hot spot, as a ratio of the peak overflow value. The software checks the congestion for the gcell by dividing the overflow value of the most over-congested gcell (that is, the peak overflow value) by the overflow value of the gcell. If the resulting congestion value is less than the specified *`cutOffRatioOfPeakHotSpot`*, the software determines this to be the boundary for that side of the hot spot, and begins checking in a new direction. *Min*: `0` *Max*: `1` *Default:* `0.01` *Data_type*: float, optional
`-cutoff_value` *`cutOffCongestNum`*	
	Specifies the minimum overflow value a gcell must have to be considered part of the hot spot. *Default:* `-2` *Data_type*: int, optional
`-mode` *`modeName`*	Specifies the mode, average or max, on which congestion hot spots in the design are reported. *Data_type*: string, optional
`-help`	Outputs a brief description that includes the type and default information for each `report_congested_area` parameter. For a detailed description of the command and all of its parameters, use the man command `man report_congested_area`.
`-num [all |` *`numOfHotSpot`*`]`	
	Specifies how many hot spots to report. *Default:* `10` • `all`: Reports all hot spots in the design. • *`numOfHotSpot`:* Reports up to the specified number of hot spots in the design. *Data_type*: string, optional
`-out_file` *`fileName`*	Outputs the hot spot report to the specified file. *Data_type*: string, optional
`-step` *`num`*	Specifies the number of gcells to check for congestion. *Default:* `3` *Data_type*: int, optional

Examples

- The following command outputs a report describing up to 10 hot spots in the design that are found by checking three gcells at a time, using an absolute congestion threshold of `-2` and a relative threshold value of `0.01`:

  ```
  report_congested_area -step 3 -cutoff_ratio 0.010000 -cutoff_value -2 -num 10 -mode average
  ```

  The resulting report contains the following hot spot information for vertical and horizontal congestion:

  - The largest overflow value for the hot spot (`#Peak`)
  - The location of the gcell that contains the largest overflow value (`Location`)
  - The boundary coordinates for the hot spot (`Boundary Box`)
  - The area of the hot spot, in gcells (`Area`)
  - The following example shows the congestion hot spot report generated by the previous command:

    ```
    # Vertical Congestion Area Report
    #Peak   Location(X,Y)      Boundary Box                 Area(#Gcell)
    -36      (1791.7,  376.4)    (1778,  332) (1792,  786)    (6x123)
    -23      ( 607.2,  284.1)    ( 580,  251) ( 607,  306)    (12x15)
    -20      (1791.7, 2236.1)    (1778, 2214) (1792, 2247)    (6x9)
    -18      ( 607.2,  125.5)    ( 593,   92) ( 607,  148)    (6x15)
    -15      ( 637.1,  380.1)    ( 630,  347) ( 720,  435)    (39x24)
    -14      (1791.7,  859.8)    (1778,  860) (1792,  982)    (6x33)

    # Horizontal Congestion Area Report
    #Peak   Location(X,Y)      Boundary Box                 Area(#Gcell)
    -92      (1777.9, 114.4)    (1750, 4) (1854, 114)         (45x30)
    -60      ( 623.3, 114.4)    ( 472, 4) ( 816, 125)         (150x33)
    ```

- The following line from the example shows that the largest overflow value for one hot spot is `-36`, the gcell that contains the overflow value is located at `1791.7 376.4`, the coordinates for the hot spot boundary are (`1778, 332`) (`1792, 786`), and the hot spot is 6 gcells by 123 gcells in size:

  ```
  -36      (1791.7,  376.4)    (1778,  332) (1792,  786)    (6x123)
  ```

Related Topics

- `write_congested_area_report`
- Using the NanoRoute Router chapter in the *Innovus Stylus Common UI User Guide*

report_route

```
report_route
[-help]
[-clock]
[-em]
[-fanout integer]
[-fill]
[-inst string]
[-multi_cut]
[-ndr]
[-net net_name]
[-net_length length]
[-patch]
[-pg_net string]
[-secondary_pg_nets {cellName1:pinName1}]
[-selected]
[-shield]
[-summary]
[-track_utilization [-area {{lx1 ly1 ux1 uy1} {lx2 ly2 ux2 uy2}...} | -view_window ]
  [-layer string] [-include_regular_routes ]]
```

Report all the routing related information of the design. The `report_route` command is used for reporting:

- Congestion
- Single-cut or multi-cut table for each layer.
- Via Usage summary
- DFM via reporting
- EM coverage
- Wire length
- Wire density
- DRC violations
- Statistics for selected nets
- Track utilization

Parameter

`-area {{lx1 ly1 ux1 uy1} {lx2 ly2 ux2 uy2}...}`	
	Specifies the coordinates of one or more area regions for reporting track utilization. For example, `report_route -track_utilization -area {299.9015 301.332 371.8345 372.4}` *Data_type*: box_list, optional
`-clock`	Reports statistics of clock nets. *Data_type*: bool, optional
`-em`	Reports EM statistics of nets. *Data_type*: bool, optional
`-fanout integer`	Specifies the minimum fanout for limit. *Data_type*: int, optional
`-fill`	Specifies the patch wire and active fill information. *Data_type*: bool, optional
`-help`	Outputs a brief description that includes type and default information for each `report_route` parameter. For a detailed description of the command and all of its parameters, use the man command: `man report_route`

`-include_regular_routes`	Reports track utilization for tracks blocked by regular wiring. **Note:** By default, the `report_route` command does not include regular wiring when computing track utilization. *Data_type*: bool, optional
`-inst` *string*	Specifies the instance name. *Data_type*: string, optional
`-layer` *layer* | *layer:layer*	Specifies the layer or layer range for which track utilization report is generated. For example: `report_route -track_utilization -layer 1:4` *Data_type*: string, optional
`-multi_cut`	Prints single-cut and multi-cut table for each layers. If you specify the DFM via list with `route_via_weight` attribute then this option *Data_type*: bool, optional
`-ndr`	Reports NDR statistics of nets. *Data_type*: bool, optional
`-net` *net_name*	Specifies the name of the net for which the report is generated. *Data_type*: string, optional
`-net_length length`	Specifies the minimum wire length of the net for reporting the net statistics. All nets whose length is more than the specified value *Data_type*: float, optional
`-patch`	Adds the patch length information to the report. *Data_type*: bool, optional
`-pg_net` *string*	Specifies the PG net name. *Data_type*: string, optional
`-secondary_pg_nets {`*cellName1*`:`*pinName1*`}`	
	Specifies the secondary pg pin information. It reports the routing length of the secondary pg pin from the grid tap point to pin. You report all the secondary pg pins and sort the results in descending order. Use model: `report_route -secondary_pg_nets {ENDCAP:BIASNW}` *Data_type*: string, optional
`-selected`	Reports statistics for selected nets only. *Data_type*: bool, optional
`-shield`	Reports shield coverage for nets routed by the NanoRoute router. *Data_type*: bool, optional
`-summary`	Specifies that just the summary table should be displayed in the report. The summary report contains: • Number of objects, such as cells, nets, and pins • Number and percentage of nets with one, two, three, four, or more terminals. • Names and numbers of primitives used. • Net length and connection length statistics • Total wire length and number of vias, per layer. *Data_type*: bool, optional
`-track_utilization`	Reports the track utilization. *Data_type*: bool, optional
`-view_window`	Report area in ui_view_box. *Data_type*: bool, optional

Example Report

- The following command reports the track utilization in the specified area for layers Metal1-Metal4 and include regular routes.

```
report_route -track_utilization -area {299.9015 301.332 371.8345 372.4} -layer 1:4 -include_regular_routes


#Calculating Track Utilization in region ( 299.90150 301.33200 371.83450 372.40000 ) for layer range: 1-4 [Metal1-Metal4] ...
#
#+--------+------------------+--------------+-----------+-------------+-------------+-----------+--------+----------------+
#| | BLOCKED BY | |
#+--------+------------------+--------------+-----------+-------------+-------------+-----------+--------+----------------+
#| Layer | # of Tracks | TrackLength | PG Mesh % | Blkg & IO % | BlockCell % | StdCell % | Nets % | Avail Tracks % |
#+--------+------------------+--------------+-----------+-------------+-------------+-----------+--------+----------------+
#| Metal1 | 127 | 9135.55 | 0.00 | 0.00 | 0.00 | 15.91 | 0.00 | 84.09 |
#| Metal2 | 109 | 7746.47 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 100.00 |
#| Metal3 | 127 | 9135.55 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 100.00 |
#| Metal4 | 109 | 7746.47 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 100.00 |
#+--------+------------------+--------------+-----------+-------------+-------------+-----------+--------+----------------+
#| TOTAL | 472 | 33764.04 | 0.00 | 0.00 | 0.00 | 4.30 | 0.00 | 95.70 |
#+--------+------------------+--------------+-----------+-------------+-------------+-----------+--------+----------------+
#Calculating Track Utilization: cpu:00:00:00, real:00:00:00, mem:771.6 MB, peak:1016.7 MB
```

- The following command will report for a summary.

```
report_route -summary

############################################################
Statistics for design DTMF_CHIP :
+-------------------+--------+
| Design Metric  | Count |
+-------------------+--------+
| Instances      |  5817 |
| Nets           |  6110 |
| Tri-State Nets |     0 |
| Degen nets     |     0 |
| Pins           | 22697 |
| I/O Terminals  |    57 |
+-------------------+-------+

Net Pin Statistics :
+----------------+------------+-------------+
| Number of      | Number of | Percentage |
| Connected Pins | Nets      | of Nets    |
+----------------+------------+-------------+
| 1              |          0 |      0.000 |
| 2              |       3470 |     56.792 |
| 3              |       1116 |     18.265 |
| 4              |        569 |      9.313 |
| 5              |        248 |      2.144 |
| 6              |        131 |      2.144 |
| 7              |         91 |      1.489 |
| 8              |         53 |      0.867 |
| 9              |         40 |      0.655 |
| >=10           |        392 |      6.416 |
+----------------+------------+-------------+

Wire Length Statistics :
+----------------+-----------------+
| Layer Name | Wire Length |
+----------------+-----------------+
| Metal1     |    59400 um |
| Metal2     |   170520 um |
| Metal3     | 15260520 um |
| Metal4     | 14931120 um |
| Metal5     |     2640 um |
| Metal6     |   340480 um |
```

```
+----------------+-----------------+
| Total      | 30764680 um |
+----------------+-----------------+

Via Count Statistics :
+----------------+---------+
| Via-cut Name | Count |
+----------------+---------+
| VIA01        |     0 |
| VIA12        |   618 |
| VIA23        |   613 |
| VIA34        |   811 |
| VIA45        |     4 |
| VIA56        |     2 |
+----------------+---------+
| Total        |  2048 |
+----------------+---------+

*** Net length and connection length statistics (cell DTMF_CHIP) ***
Total net length = 315847.915 (156940.305 158907.610)
Avg net length = 51.694 (sigma = 104.175)
Sqrt of avg square net length = 116.296
Avg connection length = 19.042 (sigma = 38.631)
Sqrt of avg square connection length = 43.069
Net and connection length distribution:
+--------------------------+------------------+--------------------+
|                          | Number of Nets | Number of Nets  |
| Length Range             | with           | with            |
|                          | Wire Length    | Fly-line Length |
+--------------------------+------------------+--------------------+
| 0.00 um - 13.20 um       |           2347 |             3733 |
| 13.20 um - 26.40 um      |           1491 |             1404 |
| 26.40 um - 39.60 um      |            620 |              374 |
| 39.60 um - 52.80 um      |            352 |              167 |
| 52.80 um - 66.00 um      |            262 |               93 |
| 66.00 um - 79.20 um      |            173 |               58 |
| 79.20 um - 92.40 um      |            125 |               41 |
| 92.40 um - 105.60 um     |            108 |               30 |
| 105.60 um - 118.80 um    |             79 |               19 |
| 118.80 um - 132.00 um    |             43 |               23 |
| 132.00 um - 145.20 um    |             42 |               19 |
| 145.20 um - 158.40 um    |             22 |                9 |
| 158.40 um - 171.60 um    |             17 |                8 |
| 171.60 um - 184.80 um    |             32 |               17 |
| 184.80 um - 198.00 um    |             17 |                8 |
| 198.00 um - 211.20 um    |             13 |                4 |
| 211.20 um - 224.40 um    |             13 |                6 |
| 224.40 um - 237.60 um    |             17 |               10 |
| 237.60 um - 250.80 um    |             24 |               11 |
| 250.80 um - 264.00 um    |             32 |               14 |
| 264.00 um - 277.20 um    |             17 |                4 |
| 277.20 um - 290.40 um    |             19 |                2 |
| 290.40 um - 303.60 um    |             29 |                2 |
| 303.60 um - 316.80 um    |             10 |                3 |
| 316.80 um - 330.00 um    |             20 |                2 |
| 330.00 um - 343.20 um    |             13 |                3 |
| 343.20 um - 356.40 um    |              8 |                1 |
| 356.40 um - 369.60 um    |              9 |                5 |
| 369.60 um - 382.80 um    |              3 |                2 |
| 382.80 um - 396.00 um    |              7 |                2 |
| 396.00 um - 409.20 um    |              6 |                1 |
| 409.20 um - 422.40 um    |              9 |                0 |
| 422.40 um - 435.60 um    |             13 |                4 |
| 435.60 um - 448.80 um    |             10 |                0 |
| 448.80 um - 462.00 um    |             10 |                2 |
| 462.00 um - 475.20 um    |              8 |                0 |
| 475.20 um - 488.40 um    |              6 |                1 |
| 488.40 um - 501.60 um    |              8 |                2 |
| 501.60 um - 514.80 um    |              9 |                1 |
| 514.80 um - 528.00 um    |              9 |                2 |
```

```
| 528.00 um - 541.20 um    |               8 |               4 |
| 541.20 um - 554.40 um    |               5 |               2 |
| 554.40 um - 567.60 um    |               4 |               2 |
| 567.60 um - 580.80 um    |               4 |               2 |
| 580.80 um - 594.00 um    |               1 |               1 |
| 594.00 um - 607.20 um    |               4 |               0 |
| 607.20 um - 620.40 um    |               2 |               1 |
| 620.40 um - 633.60 um    |               2 |               1 |
| 633.60 um - 646.80 um    |               1 |               0 |
| 646.80 um - 660.00 um    |               2 |               1 |
| 686.40 um - 699.60 um    |               1 |               0 |
| 699.60 um - 712.80 um    |               1 |               1 |
| 712.80 um - 726.00 um    |               1 |               0 |
| 726.00 um - 739.20 um    |               2 |               2 |
| 752.40 um - 765.60 um    |               1 |               0 |
| 765.60 um - 778.80 um    |               1 |               0 |
| 778.80 um - 792.00 um    |               2 |               0 |
| 805.20 um - 818.40 um    |               1 |               1 |
| 831.60 um - 844.80 um    |               1 |               0 |
| 844.80 um - 858.00 um    |               2 |               2 |
| 871.20 um - 884.40 um    |               2 |               1 |
| 937.20 um - 950.40 um    |               1 |               0 |
| 950.40 um - 963.60 um    |               1 |               1 |
| 963.60 um - 976.80 um    |               2 |               1 |
| 976.80 um - 990.00 um    |               1 |               0 |
| 990.00 um - 1003.20 um   |               1 |               0 |
| 1003.20 um - 1016.40 um  |               1 |               0 |
| 2613600.00 um - infinity |               3 |               0 |
+--------------------------+-----------------+-----------------+

Total number of long connections = 0
Io: nrCon=0
```

- The following command will print single-cut and multi-cut table for each layer and provide dfm via ratio.

```
report_route -multi_cut
```

```
+--------------------+--------------------+---------------+----------------+-----------+
| Via-Cut Layer Name |    single-cut      |   multi-cut   |      DFM       |   Total   |
+--------------------+--------------------+---------------+----------------+-----------+
|       VIA12        |                  0 |             0 |              0 |         0 |
|       VIA23        |     27975 (100.0%) |      0 (0.0%) |  18499 (66.1%) |     27975 |
|       VIA34        |      27407 (73.7%) |  9788 (26.3%) |  29894 (80.4%) |     37195 |
|       VIA45        |      23998 (62.1%) | 14625 (37.9%) |  30590 (79.2%) |     38623 |
|       VIA56        |      18885 (51.6%) | 17717 (48.4%) |  20227 (55.3%) |     36602 |
|       VIA67        |      37190 (69.0%) | 16738 (31.0%) |  17670 (32.8%) |     53928 |
|       VIA78        |      34278 (70.5%) | 14323 (29.5%) |  15770 (32.4%) |     48601 |
|       VIA89        |      19440 (56.8%) | 14806 (43.2%) |  15319 (44.7%) |     34246 |
|       VIA910       |      27544 (90.6%) |   2859 (9.4%) |   3838 (12.6%) |     30403 |
|      VIA1011       |       2207 (46.1%) |  2585 (53.9%) |   2848 (59.4%) |      4792 |
|      VIA1112       |       2299 (52.6%) |  2071 (47.4%) |   2089 (47.8%) |      4370 |
|      VIA1213       |        442 (30.9%) |   990 (69.1%) |    990 (69.1%) |      1432 |
|      VIA1314       |        193 (20.4%) |   751 (79.6%) |    751 (79.6%) |       944 |
+--------------------+--------------------+---------------+----------------+-----------+
|       Total        |    221858 (69.5%)  | 97253 (30.5%) | 158485 (49.7%) |    319111 |
```

Related Information

- route_design
- Using the NanoRoute Router chapter in the *Innovus Stylus Common UI User Guide*

report_shields

```
report_shields
[-help]
[-crosstie]
[-include_layer]
[-out_file fileName]
[-selected]
[-verbose]
```

Reports shield coverage for nets routed by the NanoRoute router.

Parameters

`-crosstie`	Reports the shield coverage of all shielded nets with crosstie attributes. *Data_type*: bool, optional
`-help`	Outputs a brief description that includes type and default information for each `report_shields` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man report_shields`.
`-include_layer`	Reports the shielding ratio per layer per net in addition to the ratio for the whole net. *Data_type*: bool, optional
`-out_file fileName`	Specifies the report file name. *Default:* *designName*`_shield.rpt` *Data_type*: string, optional
`-selected`	Reports statistics for selected nets only. *Data_type*: bool, optional
`-verbose`	Includes a list of nets that are skipped, which includes the following nets: • `FIXED` nets • `SPECIAL` nets • Nets with the `skip_routing` attribute • Nets with opens *Data_type*: bool, optional

Examples

- The following command reports for selected nets only but breaks them down by segment. The summary report will be printed in log, and detailed report will be in *myDesign_shield.rpt* file.

```
report_shields -crosstie -selected -verbose -out_file myDesign_shield.rpt
In myDesign_shield.rpt file:
NET in:
+-----------------------------+------------+------------+----------+------------+-----------+--------+--------------+
| Segment                     | Layer name | Expected # | Actual # | Percentage | Pre-route | Shield | Power:Ground |
+-----------------------------+------------+------------+----------+------------+-----------+--------+--------------+
| 0.1600 4.8960 2.9120 4.8960 | metal1     | 42         | 0        | 0%         | 0         | 0      | 0:0          |
| 0.1600 4.8960 2.9120 4.8960 | metal3     | 28         | 11       | 39%        | 0         | 11     | 0:11         |
+-----------------------------+------------+------------+----------+------------+-----------+--------+--------------+
| Total                       |            | 70         | 11       | 15%        | 0         | 11     | 0:11         |
+-----------------------------+------------+------------+----------+------------+-----------+--------+--------------+

In the log file:
#Crosstie Table:
#+------------+------------+----------+------------+-----------+--------+--------------+
#| Layer name | Expected # | Actual # | Percentage | Pre-route | Shield | Power:Ground |
#+------------+------------+----------+------------+-----------+--------+--------------+
#| metal1     | 42         | 0        | 0%         | 0         | 0      | 0:0          |
#| metal3     | 28         | 11       | 39%        | 0         | 11     | 0:11         |
#+------------+------------+----------+------------+-----------+--------+--------------+
#| Total      | 70         | 11       | 15%        | 0         | 11     | 0:11         |
#+------------+------------+----------+------------+-----------+--------+--------------+
```

- The following command prints a detailed report in *myDesign_shield_2.rpt* file without breaking down by segment. This command is called without the `-verbose` option.

```
report_shields -crosstie -selected -out_file myDesign_shield_2.rpt

In file myDesign_shield_2.rpt:
NET in:
+-------------+------------+------------+-------------+-----------+-----------+-----------------------+
| Layer name  | Expected # | Actual #   | Percentage  | Pre-route | Shield    | Power:Ground          |
+-------------+------------+------------+-------------+-----------+-----------+-----------------------+
| metal1      | 42         | 0          | 0%          | 0         | 0         | 0:0                   |
| metal3      | 28         | 11         | 39%         | 0         | 11        | 0:11                  |
+-------------+------------+------------+-------------+-----------+-----------+-----------------------+
| Total       | 70         | 11         | 15%         | 0         | 11        | 0:11                  |
+-------------+------------+------------+-------------+-----------+-----------+-----------------------+

In Log file:
#Crosstie Table:
#+------------+------------+----------+------------+-----------+--------+--------------+
#| Layer name | Expected # | Actual # | Percentage | Pre-route | Shield | Power:Ground |
#+------------+------------+----------+------------+-----------+--------+--------------+
#| metal1     | 42         | 0        | 0%         | 0         | 0      | 0:0          |
#| metal3     | 28         | 11       | 39%        | 0         | 11     | 0:11         |
#+------------+------------+----------+------------+-----------+--------+--------------+
#| Total      | 70         | 11       | 15%        | 0         | 11     | 0:11         |
#+------------+------------+----------+------------+-----------+--------+--------------+
```

Related Information

- `delete_shields`
- `route_add_shields`
- Using the NanoRoute Router chapter in the *Innovus Stylus Common UI User Guide*
  - Interpreting the Shielding Report

report_tracks

```
report_tracks
[-help]
[-prefer_only]
```

Reports track definitions for each layer. The output is in the DEF TRACKS format. This command can be used after the floorplan is created or after using the `add_tracks` command to see exactly which tracks were created.

Parameters

`-help`	Prints a brief description that includes the type and default information for each `report_tracks` parameter. For a detailed description of the command and all of its parameters, use the man command: `man report_tracks`.
`-prefer_only`	Reports track definitions only for the preferred direction of each layer. *Data_type*: bool, optional

Example

- The following command reports the track definitions for each layer:
  report_tracks

```
TRACKS Y 560 DO 2657 STEP 1120 LAYER Metal6 ;
TRACKS X 660 DO 1503 STEP 1980 LAYER Metal6 ;
TRACKS X 660 DO 2255 STEP 1320 LAYER Metal5 ;
TRACKS Y 560 DO 2657 STEP 1120 LAYER Metal5 ;
TRACKS Y 560 DO 2657 STEP 1120 LAYER Metal4 ;
TRACKS X 660 DO 2255 STEP 1320 LAYER Metal4 ;
TRACKS X 660 DO 2255 STEP 1320 LAYER Metal3 ;
TRACKS Y 560 DO 2657 STEP 1120 LAYER Metal3 ;
TRACKS Y 560 DO 2657 STEP 1120 LAYER Metal2 ;
TRACKS X 660 DO 2255 STEP 1320 LAYER Metal2 ;
TRACKS X 660 DO 2255 STEP 1320 LAYER Metal1 ;
TRACKS Y 560 DO 2657 STEP 1120 LAYER Metal1 ;
```

- The following command reports the track definitions only for the preferred direction of each layer:
  `report_tracks -prefer_only`

```
TRACKS X 660 DO 1503 STEP 1980 LAYER Metal6 ;
TRACKS Y 560 DO 2657 STEP 1120 LAYER Metal5 ;
TRACKS X 660 DO 2255 STEP 1320 LAYER Metal4 ;
TRACKS Y 560 DO 2657 STEP 1120 LAYER Metal3 ;
TRACKS X 660 DO 2255 STEP 1320 LAYER Metal2 ;
TRACKS Y 560 DO 2657 STEP 1120 LAYER Metal1 ;
```

Related Information

- `add_tracks`
- `check_tracks`
- Using the NanoRoute Router chapter in the *Innovus Stylus Common UI User Guide*
  - Adding Tracks
  - Running the NanoRoute Router with Innovus Text Commands

report_wires

```
report_wires
[-help]
[out_file]
[-detail]
[min_wire_to_half_bbox_ratio]
[-max_fanout integer]
[-min_wire_length double]
[-sort {ratio wire_length half_perimeter}]
[-summary]
```

Creates a report file that contains wire statistics for signal nets with a real wire length or minimum wire length that is greater than the specified threshold ratio. The wire statistics for signal nets can also be based on the fanout or wire length of the net.

`Ratio (r)` is calculated as `r = (total net routing length) / (half-perimeter of net bounding box)`.

The wire information reported includes the net name, the wire ratio, the real and minimum wire lengths, the number of vias, and the number of fanins and fanouts on the net. Use the `report_wires` command after routing with any of the routing commands in the Innovus software.

Note: Use the `report_special_routes` command to generate a report for special nets.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `report_wires` parameter. For a detailed description of the command and all of its parameters, use the man command `man report_wires`.
`-detail`	Includes the following additional information in the report: the wire length and number of vias for each layer, and a summary of the total wire length and via number for the design. *Data_type*: bool, optional
out_file	Creates a output report file using the specified name. *Default:* Creates a report file named `topcell.wirerpt` *Data_type*: string, optional
`-max_fanout` *fanout*	Specifies the maximum number of net's fanout. *Data_type*: int, optional
`-min_wire_length` *length*	Specifies the minimum wire length of the net. *Data_type*: double, optional
`-sort {ratio wire_length half_perimeter}`	
	Specifies the order in which the nets are sorted in the output report. The nets can be sorted as per the ratio, wire length, or half perimeter length. *Data_type*: enum, optional
`-summary`	Specifies that just the summary table should be displayed in the report. *Data_type*: bool, optional
min_wire_to_half_bbox_ratio	Reports wire statistics for wires with lengths greater than the specified threshold ratio. If you do not specify a threshold ratio, the `report_wires` command calculates wire information for wires with a ratio greater than `1.0`. *Default:* `1.0` **Note:** There are many situations in which the threshold ratio can be less than `1.0`. To ensure that the software creates a report on all nets, specify a ratio value of `0`. *Data_type*: float, optional

Examples

- The following command writes the wire statistics for all nets with wire lengths greater than the 1.0 threshold ratio:

```
report_wires 1.0
```

The following example shows a section of the wire information reported:

```
ratio      wire_length  half_perimeter  vias fan_in fan_out net_name
                           length
------------------------------------------------------------------------
1.08 Total   177.090     164.16 Total     4     1       1     dOut2[1]
1.12 Total    79.185      70.94 Total     3     1       1     dOut2[0]
1.04 Total   179.960     173.46 Total     7     1       4     dOut2[1]
1.00 Total   178.680     178.02 Total     8     1       4     dOut2[0]
1.02 Total   203.420     199.76 Total    10     1       4     dOut2[1]
1.25 Total   222.420     178.30 Total    12     1       4     dOut2[0]
1.18 Total   185.645     157.32 Total    11     1       4     s2[1]
1.09 Total   456.740     155.76 Total    10     1       4     s2[0]
...
```

- The following command writes the detailed wire statistics for all nets with wire lengths greater than the 1.0 ratio to the `detail.wirerpt` file. This report includes the wire length and number of vias for each layer, and a summary of the total wire length and via number for the design:

  ```
  report_wires -detail detail.wirerpt 1.0
  ```

  The following example shows a section of the wire information and the summary from the end of the report:

  ```
  ratio      wire_length      half_perimeter  vias fan_in fan_out net_name
                                length
  ------------------------------------------------------------------------
  1.08 Total   177.090         164.16 Total     4     1       1     dOut2[1]
       Metal1        7.650               Via01      0
       Metal2      103.440               Via12      2
       Metal3       66.000               Via23      2
       Metal4        0.000               Via34      0
       Metal5        0.000               Via45      0
       Metal6        0.000               Via56      0
  1.12 Total    79.185          70.94 Total     3     1       1     dOut2[0]
       Metal1        8.245               Via01      0
       Metal2       61.040               Via12      1
       Metal3        9.900               Via23      2
       Metal4        0.000               Via34      0
       Metal5        0.000               Via45      0
  ...
  ### Summary Report ###
  Total Metal1 Wire Length = 33.99
  Total Metal2 Wire Length = 1869.13
  Total Metal3 Wire Length = 1885.62
  Total Metal4 Wire Length = 67.76
  Total Metal5 Wire Length = 0.00
  Total Metal6 Wire Length = 0.00
  ---------------------------------
  Total Wire Length = 3856.51
  Total Via01 Number = 0
  Total Via12 Number = 272
  Total Via23 Number = 242
  Total Via34 Number = 3
  Total Via45 Number = 0
  Total Via56 Number = 0
  ---------------------------------
  ```

Related Information

- `report_special_routes`
- Using the NanoRoute Router chapter in the *Innovus Stylus Common UI User Guide*

route_add_shields

```
route_add_shields
[-help]
[-crosstie_only]
[-selected]
[-include_fixed]
```

Creates shield wires for the NanoRoute router. After creating the shields, the router generates a report that includes shielding coverage statistics.

Parameters

`-crosstie_only`	Adds crosstie for coaxial shielding only. *Data_type*: bool, optional
`-help`	Outputs a brief description that includes the type and default information for each `route_add_shields` parameter. For a detailed description of the command and all of its parameters, use the man command `man route_add_shields.`
`-include_fixed`	Includes fixed wires for shielding. The wire will not be changed but the electrical characteristics can be altered by adding shielding so that this option is required to include the fixed wire for shielding. *Data_type*: bool, optional
`-selected`	Creates shield wires for the selected nets. If you do not specify this option, all the clock nets will be shielding. *Data_type*: bool, optional

Example

- The following command creates shield wires for selected nets:
  ```
  route_add_shields -selected
  ```

Related Information

- `delete_shields`
- `report_shields`
- Using the NanoRoute Router chapter in the *Innovus Stylus Common UI User Guide*
  - Interpreting the Shield Report

route_design

```
route_design
[-help]
[-backside]
[-bump]
[-high_freq]
[-selected]
[-via_pillar_opt]
[[[-clock_eco ] [-global_detail]
[-placement_check | -no_placement_check ]
[[ -track_opt ] [-ideal_clock ]]
[-via_opt ] [-wire_opt ]] | [-passive_fill ]]
```

Runs routing or postroute via or wire optimization using the NanoRoute router. If you specify this command without any arguments, it runs global and detailed routing.

`The route_design` command has the following characteristics:

- It supports multiple-CPU processing.
- It checks for conflicts in option settings and issues warning messages when it finds problems. For example, trying to fix postroute lithography problems and optimize vias concurrently can cause conflicts.
  If `route_design` detects requests for both types of operation, it issues a warning, turns off via optimization, and proceeds with fixing lithography problems.
- It honors all route Category Attributes that do not have conflicts
- `route_design` is a super command that handles the setting of NanoRoute variables so that it is run in timing-driven mode then sets them back postRoute. It runs SMART routing by default; that is, it runs in both timing- and signal integrity-driven mode by default.
  Note: The other routing commands are not timing- or signal-integrity driven by default, but you can use the following route Category Attributes to turn on timing- and signal-integrity-driven routing for those commands:
  - To turn on timing-driven routing, run the following route category attribute before running `route_design`:
    ```
    set_db route_design_with_timing_driven 1
    ```
  - To turn on SI-driven routing, run the following route category attribute before running `route_design`:
    ```
    set_db route_design_with_si_driven 1
    ```
- If you run either (or both) of these attributes, the software stores the mode settings in the .mode file. If you restore the design and run `route_design` again, it honors the these settings. For example, if you type the following commands, the next time you restore the design and run `route_design`, it routes the design in timing-driven mode:
  ```
  set_db route_design_with_timing_driven 1
  route_design -global_detail
  write_db
  ```
- It updates the timing file automatically after all the nets are detail routed in timing-driven mode. Updating the file ensures that critical nets are not touched during postroute multiple-cut via swap, wire spreading and wire widening. When determining whether all the nets are detail routed, the router skips pins without physical ports.
- It runs a placement check prior to routing and displays a warning message if the placement is not clean.
- To turn off the placement check, specify the following parameter:
  ```
  route_design -no_placement_check
  ```
- It changes the status of clock nets from FIXED to ROUTED so it can modify them during routing and routes them before routing other nets. Once the status of the clock nets is set to ROUTED, it does not change it back to FIXED.
  - To keep clock nets' status FIXED, run the following route category attribute before running `route_design`:
    ```
    set_db route_design_fix_clock_nets 1
    ```
  - To stop the router from routing clock nets first, run the following route category attribute before running `route_design`:
    ```
    set_db route_design_route_clock_nets_first 1
    ```

Note: You can use the `check_design -type route` command to detect the potential problems and correct the design before running the `route_design` command

- For issues related to unplaced instances that clearly affect routing, the `check_design` command displays error messages and returns a status to discontinue the flow. You must resolve the issue before continuing to the route phase.
- For issues related to adherence to the track rules, the `check_design` command checks the design. It currently makes a basic check that the structures

needed for routing adhere to the tracking rules before routing begins.

- For issues related to unplaced top level pins that clearly affect routing, the `check_design` command displays error messages and returns a status to discontinue the flow. You must resolve the issue before continuing to the route phase.
- For issues related to adherence to the third party routing rules, the `check_design` command checks the design for metal violations. If the design is taped out or known to be clean, the presence of violations represents a discrepancy between the design and the tech.lef. It normally indicates that the tech may be of the wrong version or may need to be updated to the technology that the design was taped out using. These issues must be understood in detail and waved individually and on purpose by changing the tech.lef before continuing to the route phase.
- For issues related to overlapping cells that clearly affect routing, the `check_design` command displays error messages and returns a status to discontinue the flow. You must resolve the issue before continuing to the route phase.
- For issues related pin spacing violations, the `check_design` command checks if the pin has a spacing violation with the same cell or a different metal cell. It checks macro pin related spacing violations and runs the `check_drc` command to catch spacing issues that may block the pin access.You must resolve the issue before continuing to the route phase.
- For issues related to routing congestion, the `check_design` command checks the NanoRoute congestion table and reports the congestion in the NanoRoute log format (with the information that the source is NanoRoute Global). Alternatively, it reports the Early Global Route congestion in the NanoRoute format. By default, the `check_design` command stops if the Worst Over Congested Gcell is above 0.75%. Even though these issues can be ignored, it is recommended that you understand them in detail and wave them individually before continuing to the route phase.
- For issues related to a blockage between the pin and the attribute constraint layer at the pin location, the `check_design` command displays error messages and returns a status to discontinue the flow. You must resolve the issue before continuing to the route phase.
- For issues related to pre-existing violations in ECO routing, the `check_design` command runs the `check_drc` command and then checks the results for routing wire DRC and ignores non-routing wire DRC. If any metal violation is found, the `check_design` command displays error messages and returns a status to discontinue the flow. You must resolve the issue before continuing with ECO routing.
- For issues related to adherence to the third party routing rules, the check_design command checks the design for metal violations. If the design is taped out or known to be clean, the presence of violations represents a discrepancy between the design and the tech.lef. It normally indicates that the tech may be of the wrong version or may need to be updated to the technology that the design was taped out using. These issues must be understood in detail and waved individually and on purpose by changing the tech.lef before continuing to the route phase.
- For issues related to unplaced top level pins that clearly affect routing, the `check_design` command displays error messages and returns a status to discontinue the flow. You must resolve the issue before continuing to the route phase.
- For issues related to pin access for non-default rules, the `check_design` command checks each NDR nets output pin for two-cut APA accessibility. For macros, it checks if the pin width is less than or equal to the NDR. The `check_design` command displays error messages if the NDR is hard or if the no taper property is set to true. You must resolve the issue before continuing to the route phase.

Parameters

`-backside`	Routes backside layers only. The NanoRoute router supports backside routing for the layers defined as "BACKSIDE" in LEF and cell library pins. When the `-backside` parameter is enabled, the router connects the pins in the backside layers only. **Note:** Backside routing layer has the “BACKSIDE” LEF property assigned to it. The backside rule is used to specify that the routing layer is used on the underside of the die. *Data_type*: bool, optional
`-bump`	Enables interposer routing to connect the macro bumps with bus routing patterns. **Note:** Before enabling this functionality, you must specify the routing settings using the `route_interposer*` attributes in the `route` category. *Data_type*: bool, optional
`-help`	Outputs a brief description that includes type and default information for each `route_design` parameter. For a detailed description of the command and all of its parameters, use the man command: `man route_design`.
`-clock_eco`	Runs eco routing only on clock nets. *Data_type*: bool, optional
`-global_detail`	Runs timing-driven and SI-driven global and detailed routing. **Note**: `-global_detail` is the default value for this command. *Data_type*: bool, optional

`-high_freq`	Runs high frequency routing. It routes high frequency digital signals to achieve critical performance. *Data_type*: bool, optional
`-ideal_clock`	Performs timing optimization using ideal clocks instead of propagated clocks. Using ideal clocks enables you to either optimize timing without building a clock tree first, or to ignore the existing clock tree, to close ideal timing. If this parameter is not specified, the software assumes that the clock tree has been built, and optimizes timing based on propagated clocks. **Note**: The `-track_opt` parameter must be specified while using this parameter. **Note**: When this parameter is specified, all propagated clock mode settings are reset; the `timing_analysis_clock_propagation_mode` attribute is set to `forcedIdeal`. *Data_type*: bool, optional
`-no_placement_check`	Turns off placement checking. By default, `route_design` runs a placement check before routing and displays a warning message if the placement is not clean. For information on the types of checks and the violation report, see `check_place`. *Data_type*: bool, optional
`-passive_fill`	Performs passive fill. *Data_type*: bool, optional
`-placement_check`	Checks for placement violations before routing. If the placement is not clean, `route_design` display a warning message. To bypass this check use `route_design -no_placement_check`. *Data_type*: bool, optional
`-selected`	Routes selected nets only. *Data_type*: bool, optional
`-track_opt`	Performs timing optimization after track assignment. Specify this option to ensure that TQuantus model file is used during `time_design` and `opt_design -post_route` stages. **Note**: If this parameter is specified and the TQuantus model file is missing, the TQuantus model file will be generated and set automatically. If the timing library does not exist, timing optimization will be skipped and continue routing. For details of the TQuantus model file usage, see the "Creating RC Model Data in TQuantus Model File" section in the Using the NanoRoute Router chapter of the *Innovus User Guide*. *Data_type*: bool, optional
`-via_opt`	Optimizes vias after routing. **Warning**: Exercise caution with the use of this parameter, as the router might create violations during postroute via optimization. It runs a search and repair step after optimization if violations occur. *Data_type*: bool, optional
`-via_pillar_opt`	Performs opportunistic via pillars. **Note**: After the design is routed or post-route optimized, the NanoRoute router looks for available tracks around the instance pin to either add a new via pillar or upgrade the original smaller via pillar to the one with more fingers. You can perform opportunistic via pillar insertion by using the `route_design -via_pillar_opt` command. After opportunistic via pillar insertion, there is no need for route_eco; opportunistic via pillar insertion is DRC clean by construction. *Data_type*: bool, optional
`-wire_opt`	Optimizes wires after routing. This parameter, when specified, widens and shrinks wires after detail routing has been performed. The settings specified using the following route Category Attributes, impact the behavior of this parameter: • `route_design_detail_post_route_wire_widen` • `route_design_detail_post_route_spread_wire` • `route_design_detail_post_route_wire_widen_rule` *Data_type*: bool, optional

Examples

Routing

- To run timing- and SI-driven global and detailed routing, type the following command:
  ```
  route_design
  ```
- To run timing- and SI-driven global and detailed routing, type the following commands:
  ```
  route_design -global_detail
  ```
- To run non-timing-driven global and detailed routing, type the following commands:
  ```
  set_db route_with_timing_driven 0
  route_design -global_detail
  ```

Postroute Yield Optimization

- To run postroute single-cut via to multiple-cut via swap, type the following commands:
  ```
  set_db route_design_detail_min_slack_for_opt_wire slack
  set_db route_design_detail_post_route_swap_via multiCut
  route_design -via_opt
  ```
- To run non-timing-driven postroute single-cut via to multiple-cut via swap, type the following commands:
  ```
  set_db route_design_with_timing_driven 0
  set_db route_design_detail_post_route_swap_via multiCut
  route_design -via_opt
  ```
- To run postroute multiple-cut via to single-cut via swap, type the following commands:
  ```
  set_db route_design_detail_min_slack_for_opt_wire slack
  set_db route_design_detail_post_route_swap_via singleCut
  route_design -via_opt
  ```
- To run non-timing driven postroute multiple-cut via to single-cut via swap, type the following commands:
  ```
  set_db route_design_with_timing_driven 0
  set_db route_design_detail_post_route_swap_via singleCut
  route_design -via_opt
  ```
- To run postroute wire widening, type the following commands:
  ```
  set_db route_detail_min_slack_for_opt_wire slack
  set_db route_detail_post_route_wire_widen_rule ruleName
  set_db route_detail_post_route_wire_widen widen
  route_design -wire_opt
  ```
- To run non-timing driven postroute wire widening, type the following commands:
  ```
  set_db route_design_detail_post_route_wire_widen_rule ruleName
  set_db route_design_with_timing_driven 0
  set_db route_design_detail_post_route_wire_widen widen
  route_design -wire_opt
  ```
- To run postroute wire unwidening, type the following commands:
  ```
  set_db route_detail_min_slack_for_opt_wire slack
  set_db route_detail_post_route_wire_widen_rule ruleName
  set_db route_detail_post_route_wire_widen shrink
  route_design -wire_opt
  ```
- To run non-timing-driven postroute wire unwidening, type the following commands:
  ```
  set_db route_design_detail_post_route_wire_widen_rule ruleName
  set_db route_design_with_timing_driven 0
  set_db route_design_detail_post_route_wire_widen shrink
  route_design -wire_opt
  ```
- To run postroute wire spreading, type the following commands:

```
set_db route_design_detail_min_slack_for_opt_wire slack
set_db route_design_detail_post_route_spread_wire 1
route_design -wire_opt
```

- To run non-timing-driven postroute wire spreading, type the following commands:

```
set_db route_design_with_timing_driven 0
set_db route_design_detail_post_route_spread_wire 1
route_design -wire_opt
```

Related Information

- `route_detail`
- `route_global_detail`
- `route` Category Attributes
- Using the NanoRoute Router chapter in the *Innovus Stylus Common UI User Guide*
  - Using the route_design Supercommand
  - Postroute Via Optimization

route_detail

```
route_detail
[-help]
[-selected]
[area]
```

Uses the NanoRoute router to perform detailed routing on the entire design, an rect of the design specified by the bounding box, or on selected nets. If you do not specify parameters, the router routes the entire design. If you specify both the bounding box and selected nets, the router routes only selected nets within the bounding box. You can perform detailed routing on the specified rectangle only if you are rerouting the design. If you have not already run detailed routing on the entire design, you cannot run detailed routing on a specified rect.

This command supports multiple-CPU processing.

Use this command after completing the following steps:

1. Placing the design.
2. Routing special nets.
3. Setting net attributes with the `set_route_attributes` command and the `route` Category Attributes.
   Note: This step is optional. Complete this step only if you do not want to use the current values for attributes or default values for options.
4. Running global routing with the `route_design -global` command

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `route_detail` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man route_detail`
area	Specifies the x and y coordinates of the bounding box rectangle. *Data_type*: rect, optional
`-selected`	Routes selected nets. To select a net, click the net in the design display window. **Tip**: You can also set the `route_selected_net_only` attribute to `1` to route the selected nets. *Data_type*: bool, optional

Example

The following command selects `net1` and runs detailed routing on it:

```
select_obj [get_db nets net1]
route_detail -selected
```

Related Information

- `set_route_attributes`
- `route_design`
- `route` Category Attributes
- Using the NanoRoute Router chapter in the *Innovus Stylus Common UI User Guide*
  - Detailed Routing

route_early_global

```
route_early_global
[-help]
```

Early Global Route (`route_early_global`) is a quick global routing for estimating routing-related congestion and parasitic (resistance and capacitance) values. The Early Global Route results are also used for pin assignment when you commit partitions in hierarchical designs. Early Global Route does not guarantee DRC-clean routing results. Do not perform signal integrity analysis on a design that has been routed using Early Global Route, because these routes are only used to estimate parasitic values for timing analysis. Route designs with the NanoRoute router or the WRoute router if you want to perform signal integrity analysis.

Wire overlap can occur if a gcell is only partially covered by a routing blockage or pre-routed wires. Overlapping of nondefault rule wires can also occur during track assignment; however, this information is modeled when the router estimates congestion. You can use the `route_early_global` command after running placement. If the design uses additional placement features, such as clock tree synthesis and scan groups, use the command after running these placement features. If you have run IPO commands on your design, you must run the `route_early_global` command to update interconnections for the design.

Notes:

- You can set the global parameters for the Early Global Route program by using the route_early_global Category Attributes. Attributes that you specify with route_early_global Category Attributes are then used automatically whenever you run Early Global Route, either explicitly through the `route_early_global` command or internally during in-place optimization, partitioning, or placement.
- The `assign_partition_pins` command reports the number of guide pins from route guides and fixed pins. Since `route_early_global` chooses guide points provided by `assign_partition_pins` to lay wires on partition boundary, the `route_early_global` report clearly differentiates between fixed pins and the pins which are in the pin guide.
- The Early Global Router supports nested partitions. The partitions specified with the `route_early_global_honor_partition_fence` and the `route_early_global_honor_partition_pin` attributes are routed in the style that honors fence and pin constraints. All other partitions are still routed flat.
- The Early Global Router supports get_metric APIs and enables you to query existing routing metrics at command line. To enable this capability, `route_design` just calls the 'get_metrics' API after generating the routing metrics.
- If bus guides have been created in the GUI, the `route_early_global` command honors the bus guides automatically.
- The `route_early_global` command honors multiple-CPU setting for track assignment inside r`oute_early_global` by default.
- The Early Global Router supports blackboxes by default. It internally calls the `assign_partition_pins` command and places and routes all unplaced blackbox pins.
- The Early Global Router supports uses two-row-height gcells with congestion modelling improvement for the technology nodes whose row-height is smaller than 7 track. It uses 2x2 gcell size as default for small default gcell size (<=7). By default, it checks for the number of tracks rather than checking the technology of the design to grow gcell size.

  Note: This enhancement is designed only for N7 designs as these designs have less than 7 tracks. If an N7 design has more than 7 tracks then these setting will not be applied in order to ensure a better congestion accuracy.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `route_early_global` parameter. For a detailed description of the command and all of its parameters, use the man command: `man route_early_global`.

Example

- The following command honors partition fences with single entry and does not allow crossing other partitions, when `route_early_global` is run:

```
get_db route_early_global_honor_honor_fence {.}
route_early_global
```

- The following command does quick global routing for estimating routing-related congestion and parasitic values.

```
route_early_global

#% Begin route_early_global (date=02/10 16:11:30, mem=1043.0M)
[NR-eGR] Started Early Global Route kernel ( Curr Mem: 1187.03 MB )
[NR-eGR] Num Prerouted Nets = 0 Num Prerouted Wires = 0
[NR-eGR] Read 7482 nets ( ignored 0 )
[NR-eGR] Layer group 1: route 7425 net(s) in layer range [2, 6]
[NR-eGR] Early Global Route overflow of layer group 1: 0.01% H + 0.00% V. EstWL: 5.111114e+05um
[NR-eGR] Overflow after Early Global Route 0.03% H + 0.00% V
[NR-eGR] Length (um) Vias
[NR-eGR] ----------------------------------
[NR-eGR] Metal1 (1H) 0 25155
[NR-eGR] Metal2 (2V) 134127 35802
[NR-eGR] Metal3 (3H) 189844 5296
[NR-eGR] Metal4 (4V) 95825 1646
[NR-eGR] Metal5 (5H) 77137 416
[NR-eGR] Metal6 (6V) 29560 0
[NR-eGR] ----------------------------------
[NR-eGR] Total 526492 68315
[NR-eGR] ------------------------------------------------------------------------------
[NR-eGR] Total half perimeter of net bounding box: 471333um
[NR-eGR] Total length: 526492um, number of vias: 68315
[NR-eGR] ------------------------------------------------------------------------------
[NR-eGR] Total eGR-routed clock nets wire length: 0um, number of vias: 0
[NR-eGR] ------------------------------------------------------------------------------
[NR-eGR] Finished Early Global Route kernel ( CPU: 0.47 sec, Real: 0.72 sec, Curr Mem: 1225.82 MB )
#% End route_early_global (date=02/10 16:11:31, total cpu=0:00:00.5, real=0:00:01.0, peak res=1073.6M, current mem=1058.7M)
```

Related Information

- `route_early_global` Category Attributes
- Using Early Global Route for Congestion and Timing Analysis chapter in the *Innovus Stylus Common UI User Guide*

route_early_global Category Attributes

The root attributes in the `route_early_global` category are:

- route_early_global_effort_level
- route_early_global_honor_partition_allow_feedthru
- route_early_global_honor_partition_fence
- route_early_global_honor_partition_pin
- route_early_global_honor_partition_pin_guide
- route_early_global_honor_power_domain
- route_early_global_horizontal_supply_scale_factor
- route_early_global_num_tracks_per_clock_wire
- route_early_global_reverse_direction_regions
- route_early_global_route_selected_net_only
- route_early_global_secondary_pg
- route_early_global_secondary_pg_max_fanout
- route_early_global_stripe_layer_range
- route_early_global_vertical_supply_scale_factor

To set the value of any root attribute, use the `set_db` command:

```
set_db attribute_name value
```

To get the current value of any root attribute, use the `get_db` command:

```
get_db attribute_name
```

To get the current value of all attributes in the `route_early_global` category, use the following command:

```
get_db -category route_early_global
```

route_early_global_effort_level

`route_early_global_effort_level {standard | medium | low}`

Default: `standard`

Data_type: enum, read/write

Specifies the congestion effort level.

Note: The `low` effort level value will reduce the runtime while `standard` effort level will increase the congestion accuracy

- `standard`: Congestion is more accurate and runtime is larger.
- `medium`: Runtime will be much smaller and congestion report will have less accuracy.
- low: Runtime is faster than `medium` but less accurate.

Related Commands

- `place_design`
- `opt_design`
- `route_early_global`

route_early_global_honor_partition_allow_feedthru

`route_early_global_honor_partition_allow_feedthru list_of_ptn_cell_names`

Default: `""`

Data_type: string, read/write

Specifies the partition cell names for which the Early Global Router honors partition fences.

Related Commands

- `place_design`
- `route_early_global`

route_early_global_honor_partition_fence

`route_early_global_honor_partition_fence` *`list_of_ptn_cell_names`*

Default: `""`

Data_type: string, read/write

Honors partition fences. It routes a net belonging to more than one partition so that the routing does not violate partition boundaries, in addition to routing intra-partition and global nets using the specifications of the partitions. For example, if a net belongs to partitions `P1` and `P3`, but not `P2`, the net is routed without going over the `P2` partition boundary.

Notes:

- For flat partitioned designs, you can specify the `route_early_global_honor_partition_fence` attribute to simulate channel-based routing. For channel-less designs, you must perform feedthrough insertion before using this parameter.
- Supports designs with up to 180 partitions.

Related Commands

- `place_design`
- `route_early_global`

route_early_global_honor_partition_pin

`route_early_global_honor_partition_pin` *`list_of_ptn_cell_names`*

Default: `""`

Data_type: string, read/write

Honors partition fences with single-entry constraint and pre-assigned pins (pins marked `FIXED`) and assigned pins (pins marked `PLACED`).

Related Commands

- `place_design`
- `route_early_global`

route_early_global_honor_partition_pin_guide

`route_early_global_honor_partition_pin_guide {true|false}`

Default: `true`

Data_type: bool, read/write

Honors partition pin guides. It uses the pin guide statements in the floorplan file to guide the routing through partition pin points.

Note: Fixed pins and pin guides are only honored when the `route_early_global_honor_partition_pin_guide` attribute is `true`.

Related Commands

- `place_design`
- `opt_design`
- `route_early_global`

route_early_global_honor_power_domain

`route_early_global_honor_power_domain {true|false}`

Default: `false`

Data_type: bool, read/write

Honors MSV domain constraints. This option is used to decide whether Early Global Route should honor power domain settings.

By default, Early Global Route ignores the power domain settings and routes as flat design. When enabled, Early Global Route honors power domain settings and route wires inside preferred power domain as much as possible.

Related Commands

- `place_design`
- `opt_design`
- `route_early_global`

route_early_global_horizontal_supply_scale_factor

`route_early_global_horizontal_supply_scale_factor` *`value`*

Default: `1`

Max: `1.250000`

Min: `0.750000`

Data_type: float, read/write

Specifies the Scale H supply by a specified ratio.

Related Commands

- `place_design`
- `opt_design`
- `route_early_global`

route_early_global_num_tracks_per_clock_wire

`route_early_global_num_tracks_per_clock_wire` *`value`*

Default: `0`

Max: `8`

Min: 0

Data_type: int, read/write

Specifies the special number of tracks per clock wires.

Related Commands

- `place_design`
- `opt_design`
- `route_early_global`

route_early_global_reverse_direction_regions

`route_early_global_reverse_direction_regions "(`*`x1 y1 x2 y2`*`) Metal1:Metal2"`

Default: ""

Data_type: string, read/write

Reverses the routing direction in the given region on the specified layer-range. It specifies an area in which The Early Global Router routes wires in the non-preferred direction.

Related Commands

- place_design
- opt_design
- route_early_global

route_early_global_route_selected_net_only

```
route_early_global_route_selected_net_only {true | false}
```

Default: `false`

Data_type: bool, read/write

Specifies that only the nets that are selected in the database are to be routed.

Related Commands

- route_early_global

route_early_global_secondary_pg

```
route_early_global_secondary_pg {true | false}
```

Default: `false`

Data_type: bool, read/write

Specifies whether the secondary PG pin should be routed or not

Related Commands

- place_design
- opt_design
- route_early_global

route_early_global_secondary_pg_max_fanout

```
route_early_global_secondary_pg_max_fanout integer
```

Default: `2147483647`

Max: `INT_MAX`

Min: `1`

Data_type: int, read/write

Specifies the maximum fanout value during secondary PG pin routing.

Related Commands

- place_design
- opt_design
- route_early_global

route_early_global_stripe_layer_range

```
route_early_global_stripe_layer_range {layer_x1:layer_x2}
```

Default: " "

Data_type: string, read/write

Specifies the stripe layer range that is routed to the secondary PG pin.

Related Commands

- `place_design`
- `opt_design`
- `route_early_global`

route_early_global_vertical_supply_scale_factor

`route_early_global_vertical_supply_scale_factor` *value*

Default: `1`

Max: `1.250000`

Min: `0.750000`

Data_type: float, read/write

Specifies the Scale V supply by a specified ratio.

Related Commands

- `place_design`
- `opt_design`
- `route_early_global`

route_eco

```
route_eco
[-help]
[-ignore_route string]
[-handle_partition ]
[-fix_drc [-cut_color_flip {0 1}]
[-trim_layer_patch ] [-layer_range bottomLayer:topLayer] [area]]
[-fix_filler_drc_with_patch_only [-passive_fill ] [-layer_range bottomLayer:topLayer] [area]]
[ -fix_filler_drc_with_patch_only | -fix_drc ]
[-target | -fix_drc | -prototype ]
[-route_only_layers bottomLayer:topLayer | -fix_drc ]
```

The `route_eco` command is part of an ECO flow process and is based on the NanoRoute® router. Based on information from the `eco_read_def` command, this command preprocesses the design for the NanoRoute router and launches ECO routing within the router. It retrieves the router's mode settings and returns them to their original state at the end of the ECO routing operation.

- When you specify two layers to modify, the first number representing the bottom layer should be smaller than the second number representing top layer (for example, `-route_only_layers 2:3`), `route_eco` checks for pin accessibility for new or modified nets and drops `VIA12`, as necessary.
  Note: The `via12` and `via34` will be used, but tool will not change the metal shapes on M1 and M4.
- When a net is modified, some parts of the wire must be reassigned to the new nets. Initially, a whole wire between two cells is assigned to `netA`. When a buffer is inserted on `netA`, `route_eco` cuts the wire segment and assigns the left side of the wire to `netA` and the remaining segment on the right to `netB`.
- You must first run the `place_eco` command before you use this command.
- You can run ECO routing in multiple-CPU processing mode.

Parameters

area	Fixes DRC violations is the specified route area. You can specify the area using the *x1 y1 x2 y2* coordinate values **Note**: The parameter must be used with the `-fix_drc` parameter. *Data_type*: rect, optional
`-cut_color_flip {0 1}`	Fixes cut color DRC violations by flipping. When enabled (1), the `route_eco` command fixes the cut same-mask s **Note**: The parameter must be used with the `-fix_drc` parameter. *Data_type*: enum, optional
`-fix_drc`	Fixes violations on existing DRC markers in the design. When enabled, it allows NanoRoute to fix only the DRC's the `route_eco` command reverts to the original snap shot of the design. **Note:** If open nets are found in the design, NanoRoute will error out. **Note:** This parameter does not support the `-route_only_layers` arguments. *Data_type*: bool, optional
`-fix_filler_drc_with_patch_only`	Fixes filler related drc with metal patch only. • If only the filler-related metal patch drc exists after filler insertion, it can be fixed with the `-fix_filler_drc_with_` • The recommendation flow for chip-finish is: 1. `route_eco`. For changed cell eco 2. `add_fillers`. Inserts full fillers, even if decap and filler insertion is performed in single or multiple steps. 3. `route_eco -fix_filler_drc_with_patch_only` **Note:** Some decap cells might be replaced with normal filler cells due to handling of the vertical stack rule during a may still be left after full filler insertion if router is called after only decap cells are inserted. Hence, Cadence recom *Data_type*: bool, optional
`-help`	Outputs a brief description that includes the type and default information for each `route_eco` parameter. For a detai the man command: `man route_eco`

`-handle_partition`	Turns on handling partition flag. For the nets in hierarchical partition design, the ECO routing performed is partitior • Pure top level nets should be routed only in top area. • Inter-partition nets should be routed based on partition pin locations, and there are no multiple entries to a conn partition connection. • Intra-partition nets should be routed inside corresponding partition fence. *Data_type*: bool, optional

`-ignore_route` *string*	When specified, the selected net will be routed with highest priority by ignoring existing routes in post-route mode.
	• `all`: Ignores all other nets in the design that are not selected, including clock, secondary PG, and signal nets.
	• `signal`: Ignores only other signal nets that are not selected, but honors clock and secondary PG nets.
	For example, the following commands select and route a list of nets by ignoring only other signal nets, and consid second `route_eco` is run to fix DRC violations:
	`select_obj [get_db nets {net1 net2 net3}]`
	`set_db route_selected_net_only true`
	`route_eco -ignore_route signal`
	`reset_db -category route`
	`set_db route_selected_net_only`
	`verify_drc -limit 0`
	`route_eco`
	Default: " "
	Type: string, optional

`-layer_range` *bottomLayer:topLayer*	Enables the NanoRoute router to fix violations on existing DRC markers in the specified layer range (*bottomLayer* *Type:* string, optional
`-passive_fill`	When specified, it does passive fill after filler fixing. This helps to save extra import/export time. *Data_type*: bool, optional
`-prototype`	Enables prototyping while doing ECO routing. When specified, it enables NanoRoute to work on the nets selected snapshot for the designer. This mode is intended to help the designer know if the routing is possible and if the EC **Note:** In this mode the routes are not DRC clean. *Data_type*: bool, optional
`-route_only_layers` *bottomLayer:topLayer*	
	Enables the NanoRoute router to remove segments of net routing in the specified layer range, from `bottomLayer` to `bottomLayer` and `topLayer` values. This option also restricts eco routing to carry out only on eco nets. *Default:* ECO routing can occur on any layer and any nets. **Note:** You might want to use `-route_only_layers` to specify all layers in order for `route_eco` to avoid touching non fewer DRCs. This command is similar to the following route Category Attributes: • `route_with_eco` • `route_selected_net_only` **Note:** The NanoRoute router does not support the `-route_only_layers` option for 10nm and below nodes and disp **Note:** The `-route_only_layers` parameter cannot be specified with the `-fix_drc` parameter. **Note:** The recommended methods to know the layers and vias that were touched in the freeze metal routing with - • Use the `compare_def -metal_shape_only` command to compare the def file before and after using the `route_ec` • Use Pegasus to compare the gds file. *Data_type*: string, optional
`-target`	Enables NanoRoute to work on only eco nets identified by NanoRoute to route. In this mode, the router ignores the nets or are on nets selected to route. *Data_type*: bool, optional

`-trim_layer_patch`	Enables the `route_eco` command to only add trim and patch metals in a given area. Using the `-trim_layer_patch` and only add trim and patch metals where minimum spacing criteria is not met. *Data_type*: bool, optional

Examples

The following example allows the NanoRoute router to perform ECO routing only on or between layers 2 and 3, VIA12 will be dropped if necessary:

```
route_eco -route_only_layers 2:3
```

If DRCs exist after you run the `route_eco` command, this could be because `route_eco` cannot touch non-eco nets. You can reroute the DRC area as shown below:

```
set_db route_with_eco true
route_eco -route_only_layers 1:3
set_db route_selected_net_only false
route_global_detail 100.0 1200.0 350.0 600.0
```

Related Information

- `place_eco`
- `eco_read_def`
- Eco Flows chapter in the *Innovus Stylus Common UI User Guide*
- Accelerating the Design Process By Using Multiple-CPU Processing chapter in the *Innovus Stylus Common UI User Guide*
- Tools Menu chapter in the *Innovus Stylus Common UI Menu Reference*
  - Multiple CPU Processing

route_fix_signoff_drc

```
route_fix_signoff_drc
[-help]
[-fix_rule rule_name | -exclude_rule rule_name]
```

Fixes sign-off DRC. It fixes third party DRC violations which cannot be detected by the Innovus verification tool (like verify_drc). The third party markers need to be converted with the `read_markers` command. The supported drc types include, spacing, mincut, colorLoop, area, minhole, maxstackedvia, enclosure, width, and minstep.

Parameters

`-exclude_rules rule_name`	Specifies the sign_off tool rule type to exclude. *Data_type*: string, optional
`-fix_rules rule_name`	Specifies the sign_off tool rule type to fix. *Data_type*: string, optional
`-help`	Outputs a brief description that includes the type and default information for each `route_fix_signoff_drc` parameter. For a detailed description of the command and all of its parameters, use the man command `man route_fix_signoff_drc .`

Related Information

- `read_markers`
- Using the NanoRoute Router chapter in the *Innovus Stylus Common UI User Guide*

route_global_detail

```
route_global_detail
[-help]
[area]
[-selected]
```

Uses the NanoRoute router to perform both global and detailed routing with one command. When you load a partially routed database, the router completes the routes. This command supports multiple-CPU processing.

Note: To perform detailed routing only, use the `route_detail` command.

Use this command after completing the following steps:

1. Placing the design.
2. Routing special nets.
3. Setting net attributes with the `set_route_attributes` command and route Category Attributes.

Note: To use NanoRoute current values for attributes, and default values for options, this step is optional.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `route_global_detail` parameter. For a detailed description of the command and all of its parameters, use the `man` command: man `route_global_detail`
area	Specifies the coordinates (*x1 y1 x2 y2*) of the bounding box rect for detailed routing. The entire design is globally routed. *Data_type*: rect, optional
`-selected`	Routes selected nets. To select a net, click the net in the design display window. **Tip**: You can also set the `route_selected_net_only` attribute to `1` to route the selected nets. *Data_type*: bool, optional

Note: If you have not already run detailed routing on the entire design, you cannot run detailed routing on a specified rect.

Example

The following commands run global and detailed routing, save the design as `droute`, and generate a routed DEF file named `droute.def`.

```
route_global_detail
write_db droute
write_def -floorplan -routing droute.def
```

Related Information

- `route` Category Attributes
- `route_detail`
- `set_route_attributes`
- Using the NanoRoute Router chapter in the *Innovus Stylus Common UI User Guide*
  - Routing Phases

route Category Attributes

The root attributes in the `route` category are:

- extract_keep_fill_wires
- route_add_antenna_inst_prefix
- route_adjust_auto_via_weight
- route_allow_inst_overlaps
- route_allow_pin_as_feedthru
- route_antenna_cell_name
- route_antenna_diode_insertion
- route_concurrent_minimize_via_count_effort
- route_connect_to_bumps
- route_detail_add_passive_fill_only_on_layers
- route_detail_allow_passive_fill_only_in_layers
- route_detail_antenna_eco_list_file
- route_detail_auto_stop
- route_detail_check_mar_on_cell_pin
- route_detail_end_iteration
- route_detail_fix_antenna
- route_detail_fix_antenna_on_secondary_pg_nets
- route_detail_fix_antenna_with_gate_array_filler_mode
- route_detail_merge_abutting_cut
- route_detail_min_length_for_spread_wire
- route_detail_min_length_for_widen_wire
- route_detail_min_slack_for_opt_wire
- route_detail_no_taper_in_layers
- route_detail_no_taper_on_output_pin
- route_detail_on_grid_only
- route_detail_post_route_litho_repair
- route_detail_post_route_spread_wire
- route_detail_post_route_swap_via
- route_detail_post_route_via_pillar_effort
- route_detail_post_route_wire_widen
- route_detail_post_route_wire_widen_rule
- route_detail_postroute_via_priority
- route_detail_search_and_repair
- route_detail_signoff_effort
- route_detail_stub_routing_in_first_layer
- route_detail_use_multi_cut_via_effort
- route_diode_insertion_for_clock_nets
- route_eco_ignore_existing_route
- route_enable_route_rule_si_limit_length
- route_enforce_route_rule_on_special_net_wire
- route_extra_via_enclosure

- route_fix_clock_nets
- route_high_freq_constraint_groups
- route_high_freq_match_report_file
- route_high_freq_num_reserved_layers
- route_high_freq_remove_floating_shield
- route_high_freq_search_repair
- route_high_freq_shield_trim_length
- route_honor_exclusive_region
- route_honor_power_domain
- route_ignore_antenna_top_cell_pin
- route_ignore_follow_pin_shapes
- route_interposer_allow_diagonal_trunk
- route_interposer_interlayer_shielding_layers
- route_interposer_interlayer_shielding_nets
- route_interposer_interlayer_shielding_offsets
- route_interposer_interlayer_shielding_widths
- route_interposer_same_layer_shielding_net
- route_interposer_same_layer_shielding_width_spacing
- route_interposer_trunk_routing_layers
- route_interposer_trunk_routing_width_spacing
- route_number_fail_limit
- route_number_thread
- route_number_warning_limit
- route_process_node
- route_rc_extraction_corner
- route_relaxed_route_rule_spacing_to_power_ground_nets
- route_reserve_space_for_multi_cut
- route_reverse_direction
- route_route_clock_nets_first
- route_selected_net_only
- route_shield_crosstie_offset
- route_shield_report_skip_status
- route_shield_tap_cell_insertion
- route_shield_tap_cell_name
- route_skip_analog
- route_strictly_honor_1d_routing
- route_strict_honor_route_rule
- route_stripe_layer_range
- route_third_party_data
- route_tieoff_to_shapes
- route_trim_pull_back_distance_from_boundary
- route_trunk_with_cluster_target_size
- route_use_auto_via

- route_via_weight
- route_with_eco
- route_with_litho_driven
- route_with_si_driven
- route_with_timing_driven
- route_with_trim_metal
- route_with_via_in_pin
- route_with_via_only_for_block_cell_pin
- route_with_via_only_for_stdcell_pin
- route_trial_max_print_ignored_pad_nets

To set the value of any root attribute, use the `set_db` command:

```
set_db attribute_name value
```

To get the current value of any root attribute, use the `get_db` command:

```
get_db attribute_name
```

To get the current value of all attributes in the `route` category, use the following command:

```
get_db -category route
```

extract_keep_fill_wires

`extract_keep_fill_wires {true|false}`

Default: `false`

Data_type: bool, read/write

When enabled, the NanoRoute Router imports metal fill wires so that RC extraction can calculate them.

Related Commands

- `route_detail`
- `route_global_detail`
- `route_design`

route_add_antenna_inst_prefix

`route_add_antenna_inst_prefix value`

Default: `" "`

Data_type: string, read/write

Specifies the prefix that is added in the diode inst names. These diodes are added to fix antennas during routing.

Related Commands

- `route_detail`
- `route_global_detail`
- `route_design`

route_adjust_auto_via_weight

`route_adjust_auto_via_weight {true | false}`

Default: `true`

Data_type: bool, read/write

Allows auto tuning of internal via weight. It adjusts auto via weight for via swapping.

- When set to `true`, it enables NanoRoute to auto tune internal via weight for double cut via ratio improvement.
- When set to `false`, it disables NanoRoute to auto tune internal via weight and the via weight specified using the `route_via_weight` attribute is honored for via swapping..

Note: NanoRoute does not swap the existing dcut via's to DFM via's.

Related Commands

- `route_detail`
- `route_global_detail`
- `route_design`

route_allow_inst_overlaps

`route_allow_inst_overlaps {true | false}`

Default: `true`

Data_type: bool, read/write

Specifies that instance overlap check will be skipped when this attribute is set to `true`.

Related Commands

- `route_detail`
- `route_global_detail`
- `route_design`

route_allow_pin_as_feedthru

`route_allow_pin_as_feedthru {true|TRUE|false|FALSE|none|NONE|output|input|inout, bottomLayerNum:topLayerNum}`

Default: `true`

Data_type: enum, read/write

Specifies that pins are allowed as feedthrough within the specified layer range.

- `true`: Specifies that input/output/inout pins can be used as feedthrough.
- `false`: Specifies that output/inout pins can be used as feedthrough.
- `none`: Specifies that no pin is allowed as feedthrough.
- `output`: Specifies that output pins can be used as feedthrough.
- `input`: Specifies that input pins can be used as feedthrough.
- `inout`: Specifies that all inout pins can be used as feedthrough.
- `bottomLayerNum:topLayerNum`: Specifies the layer range that allows pins as feedthroughs.

For example, the following command allows feedthrough for M1 and M2 pins only.

```
set_db route_allow_pin_as_feedthrough "true, 1:2"
```

Related Commands

- `route_detail`
- `route_global_detail`
- `route_design`

route_antenna_cell_name

`route_antenna_cell_name {"cell_name" | "list_of_cell_names"}`

Default: `{} (empty string)`

Data_type: string, read/write

Specifies antenna diode cells to use during postroute optimization. The cells must have the same LEF `SITE` definition as the standard cells. Use this attribute with the `route_diode_insertion` attribute.

Note: If you do not specify a cell, the router uses a core macro of type ANTENNACELL specified in the LEF MACRO statement.

Related Commands

- `route_detail`
- `route_global_detail`
- `route_design`

route_antenna_diode_insertion

`route_antenna_diode_insertion {true | false}`

Default: `false`

Data_type: bool, read/write

Inserts and places antenna diode cells if there are available placement locations. By default, the NanoRoute router repairs antenna violations by changing layers (also called antenna stapling or layer hopping), but it can also repair antenna violations by inserting diode cells as close as possible to input gates to discharge current. The router inserts antenna diode cells even if filler cells are already placed. It can swap filler cells with antenna diode cells and fill the gap automatically if an antenna diode cell is not the same size as the filler cell it replaced.

To specify the diode cells, use the `route_antenna_cell_name` attribute.

Related Commands

- `route_detail`
- `route_global_detail`
- `route_design`

route_concurrent_minimize_via_count_effort

`route_concurrent_minimize_via_count_effort value`

Default: `medium`

Data_type: string, read/write

Specifies the effort level for reducing the total number of vias during routing. Use this attribute together with the `route_detail_use_multi_cut_via_effort` attribute to control the effort toward reducing the total number of vias and the effort toward increasing the ratio of double-cut vias to single-cut vias concurrently with routing.

If you specify `route_detail_use_multi_cut_via_effort medium` or `high`, the software sets the value of `route_concurrent_minimize_via_count_effort` to medium unless you specify `route_concurrent_minimize_via_count_effort` yourself. To realize further savings, set `route_concurrent_minimize_via_count_effort` to high. This could further reduce the total via count up to 10 percent.

Note: If you specify the `flow_effort` attribute of the `init Category Attributes` to `high`, the software sets the value of `route_concurrent_minimize_via_count_effort` to `high`.

Note: Results might vary with your design.

Specify one of the following attributes:

- `low`: Reduces the total via count using low effort.
- `medium`: Reduces total via count by 15 percent, with negligible runtime overhead. This is the default setting.
- `high`: Reduces the total via count by 20 percent to 25 percent, with marginal runtime overhead.

Related Commands

- route_detail
- route_global_detail
- route_design

route_connect_to_bumps

```
route_connect_to_bumps {true | false}
```

Default: `false`

Data_type: bool, read/write

Enables bump to bump routing.

Related Commands

- route_detail
- route_global_detail
- route_design

route_detail_add_passive_fill_only_on_layers

```
route_detail_add_passive_fill_only_on_layers layers
```

Default : `""`(empty string)

Data_type: string, read/write

Specifies the layers where passive fill will be added.

Related Commands

- route_detail

route_detail_allow_passive_fill_only_in_layers

```
route_detail_allow_passive_fill_only_in_layers layers
```

Default : `false`

Data_type: string, read/write

When specified, allows passive fill only in layers.

Related Commands

- route_detail

route_detail_antenna_eco_list_file

```
route_detail_antenna_eco_list_file file_name
```

Default : `{}`

Data_type: string, read/write

Specifies the file generated during process antenna repair that contains the list of diodes that were inserted. If you insert diodes during more than one detailed routing pass, the NanoRoute router creates a separate file for each pass and gives it a unique name. For example, if you use the default filename, the router creates the following files:

- After the first detailed routing pass: `nano_ant_diode.list`
- After the second detailed routing pass: `nano_ant_diode.list1`

- After the third detailed routing pass: `nano_ant_diode.list2`

Related Commands

- `route_detail`
- `route_global_detail`
- `route_design`

route_detail_auto_stop

`route_detail_auto_stop {true | false}`

Default: `true`

Data_type: bool, read/write

Controls whether NanoRoute continues routing if there are many violations. In highly congested designs, NanoRoute stops if the number of violations is too high. Specify false to continue routing until there is no improvement.

Note: The `route_detail_auto_stop` attribute is set to `0` by default for 20nm designs and below.

Related Commands

- `route_detail`
- `route_global_detail`
- `route_design`

route_detail_check_mar_on_cell_pin

`route_detail_check_mar_on_cell_pin {true|false}`

Default: `false`

Data_type: bool, read/write

Enables mar checking on used cell pins.

Related Commands

- `route_detail`
- `route_global_detail`
- `route_design`

route_detail_end_iteration

`route_detail_end_iteration` *`pass_number`*

Default: 0

Data_type: int, read/write

Specifies the last pass in a detailed routing step. To run postroute optimization and repair remaining violations, including process antenna violations, set the attribute to its default value using the following command:
`reset_db route_detail_end_iteration`

When set to its default value, the router dynamically decides when to stop. However, when you specify `route_detail_end_iteration 0`, the end iteration will be at 0.

Note: Specifying a higher value than 20 for `route_detail_end_iteration` generally does not improve results.

Related Commands

- `route_detail`

- route_global_detail
- route_design

route_detail_fix_antenna

```
route_detail_fix_antenna {true | false}
```

Default: `true`

Data_type: bool, read/write

Repairs process antenna violations by jumping layers (antenna stapling). The router repairs process antenna violations if it can do so without creating DRC violations. The router might need to make several passes before repairing all antenna violations. Specify `true` to fix antenna violation on clock net during ccopt and routeDesign (clock eco).

Note: You must set this option to true explicitly. The default value is true if you have antenna information defined in the LEF, but the router fixes antenna violations only on data nets during `route_design`, `route_global_detail`, or `route_detail`.

Related Commands

- route_detail
- route_global_detail
- route_design

route_detail_fix_antenna_on_secondary_pg_nets

```
route_detail_fix_antenna_on_secondary_pg_nets {true|false}
```

Default: `false`

Data_type: bool, read/write

Fixes the antenna violation on secondary pg nets.

Related Commands

- route_detail
- route_global_detail
- route_design

route_detail_fix_antenna_with_gate_array_filler_mode

```
route_detail_fix_antenna_with_gate_array_filler_mode  {true | false}
```

Default: `false`

Data_type: bool, read/write

When enabled, the NanoRoute router detects the GA filler flow and uses the GA filler diode cells for antenna violation fixing.

Related Commands

- route_detail
- route_global_detail
- route_design

route_detail_merge_abutting_cut

```
route_detail_merge_abutting_cut {true|false}
```

Default: `true`

Data_type: bool, read/write

Allows you to enable or disable the capability to merge cell cut shapes. When enabled, it merges two abutting cell cut shapes into a new one.

Related Commands

- route_detail
- route_global_detail
- route_design

route_detail_min_length_for_spread_wire

```
route_detail_min_length_for_spread_wire { layer1 length1 ... layerN lengthN }
```

Default: 0

Data_type: string, read/write

Specifies the minimum length in microns for a wire segment of a net before the necessity for a jog to avoid OPC issues. Spreading wires further apart (creating jogs in the wires) helps avoid design rule violations, signal integrity problems, and yield loss caused by shorts in the wires.

This parameter supports the following values:

- layer: Specifies the metal layer. The supported metal layer name is Mx, they are Metal1, Metal2, Metal3, etc.
- length: Specifies the minimum length, in microns. The length value can be a floating number, default, or skip.
  - The default value is 0, it means wire spreading will determine the threshold automatically.
  - The skip value means there is no wire spreading on this layer.

Note: You should specify the value for each metal layer. To get the number of metal layers used in the design, use db command (dbGet head.layers.name).

Note: Turn wire spreading on or off with route_detail_post_route_spread_wire

For example, when you specify the following command, no wire spreading is done on layer Metal1. For layer Metal2, the wire spreading value is set to 0 signifying that wire spreading will determine the threshold automatically. Similarly, for layers Metal3, Metal4, Metal5, and Metal6 the wire spreading length is set to 2.0.

```
set_db route_detail_min_length_for_spread_wire {Metal1 skip Metal2 0 Metal3 2.0 Metal4 2.0 Metal5 2.0 Metal6 2.0 }
```

Related Commands

- route_detail
- route_global_detail
- route_design

route_detail_min_length_for_widen_wire

```
route_detail_min_length_for_widen_wire value
```

Default: 1.0

Data_type: double, read/write

Specifies the minimum length in microns for a wire segment of a net before the segment is a candidate for widening. Wires are widened by an amount defined by a nondefault rule. They are not widened unless they meeting the following criteria:

- Timing is not adversely affected
- Routing resources are available
- No overflow is created
- No design rule or process antenna violations are created

Widening wires helps avoid design rule violations, signal integrity problems, and yield loss.

Note: To enable this attribute, you must also include a wide wire nondefault rule in the LEF file and specify the route_detail_post_route_widen_wire_rule 1 attribute.

Related Commands

- route_detail
- route_global_detail
- route_design

route_detail_min_slack_for_opt_wire

```
route_detail_min_slack_for_opt_wire value
```

Default: 0.0

Data_type: double, read/write

Specifies the minimum slack on pins associated with nets that are candidates for wire spreading or wire widening. Using this parameter reduces the timing impact of wire spreading, wire widening, and via swapping, and helps reduce congestion and the yield loss due to shorts and opens.

Note: To enable this attribute, you must also specify `route_with_timing_driven 1`.

Wire spreading and wire widening use the minimum slack value differently. While wire spreading only works for timing critical nets where the worst setup slack is less than the minimum slack value, wire widening only works for non-timing critical nets where the worst setup slack is greater than the minimum slack value.

You can use the `route_detail_min_slack_for_opt_wire` attribute to reduce the timing closure effort during post-route ECO. NanoRoute will replace via to multi-cut via on non-critical nets, if a violation does not occur.

Related Commands

- route_detail
- route_global_detail
- route_design

route_detail_no_taper_in_layers

```
route_detail_no_taper_in_layers "bottom_layer_number:top_layer_number"
```

Default : " "

Data_type: string, read/write

Prohibits the detailed router from tapering in the specified layer range, creating non-default-width wires in that range.

bottom_layer_number:top_layer_number signifies the layer range format.

Related Commands

- route_detail
- route_global_detail
- route_design

route_detail_no_taper_on_output_pin

```
route_detail_no_taper_on_output_pin {true | false | auto}
```

Default: false

Data_type: enum, read/write

Prohibits the detailed router from tapering at standard cells, macro cells, and block output pins.

Note: The default value of `route_detail_no_taper_on_output_pin` is `auto` for 32nm and below nodes.

Related Commands

- route_detail

- route_global_detail
- route_design

route_detail_on_grid_only

```
route_detail_on_grid_only none | false | true | all | {[wire [metall:metalh]] | [via [metall:metalh]]}
```

Default: `none`

Data_type: string, read/write

Controls the routing of the objects, wires, and vias in relation to the user-defined routing tracks.

- On grid wire means that the center of the wire is on the preferred routing track.
- On grid via means that the center of the via is on the intersection of preferred routing tracks.

Note: A preferred routing track is the track which has the same direction as that of the metal routing direction defined in the LAYER section in LEF.

- When there are off-grid objects on fully-routed data, NanoRoute skips those nets and issues a warning message.
- When there are fixed off-grid objects, NanoRoute skips those nets and issues a warning message.
- When there are non-fixed off-grid objects, NanoRoute deletes those nets and issues a warning message.

Specify one of the following arguments:

- `all`: Specifies that all routing objects (vias and wires) have to be on the preferred and non-preferred routing tracks. It forces all routing objects to be on grid.
- `none`: Specifies that the routing objects (vias and wires) can be off-grid. It does not force routing objects to be on grid . They can go off-grid depending on other restrictions.
- `via [metall:metalh]`: Specifies that the center of the vias must be at the intersection of preferred routing tracks. If `metall` and `metalh` are specified, all the vias between `metall` and `metalh` must be at the intersection of preferred routing tracks.
  For example, via [1:3] means all via12 must be at the intersection of preferred routing tracks of layer Metal1 and Metal2 and all via23 must be at the intersection of preferred routing tracks of layer Metal2 and Metal3.
- `wire [metall:metalh]`: Specifies that the center of the wire must be on preferred routing tracks. When a wire makes a turn on the non-preferred direction, the center of the wire might not fall on any routing tracks. If `metall` and `metalh` are specified, all the center of the wires in between those metal layers `ml` and `mh` must be on the preferred routing tracks.
  For example, wire [1:3] means the center of the Metal1, Metal2, and Metal3 wires must be on the preferred routing tracks of Metal1, Metal2, and Metal3.
- `false`: For backward compatibility purpose, the `route_detail_on_grid_only` argument also takes the value of `false`. This is equivalent to `'none'`.
- `true`: For backward compatibility purpose, the `route_detail_on_grid_only` argument also takes the value of `true`. This is equivalent to `'all'`.

Note: Block pins can be accessed by planar off-track wires.

Related Commands

- route_detail
- route_global_detail
- route_design

route_detail_post_route_litho_repair

```
route_detail_post_route_litho_repair {true | false}
```

Default: `false`

Data_type: bool, read/write

Corrects bridging, necking, and line-end shortening problems caused by lithography. Takes the severity of the problems into account and fixes the high-severity problems first. Prior to running the router with this parameter, import a hotspot interface format (HIF) file with the `read_markers` command .

Related Commands

- route_detail

- route_global_detail
- route_design

route_detail_post_route_spread_wire

```
route_detail_post_route_spread_wire  {auto| true | false | 0 | 0.25| 0.5 | 1 }
```

Default: auto

Data_type: string, read/write

Spreads routing, post route, by the percentage of the track. The router moves the wires without moving the vias. Specifying false for this attribute signifies no spreading, while specifying true signifies 0.5 / 1 track spreading by default for all nodes. The use of `true` and `false` is not needed as a value greater than `0` and less than equal to `1` (0<x=<1) activates the spreading. Specifying this attribute will not increase the overall routing runtime much but helps avoid yield loss caused by shorts and improve the timing as well.

Specify one of the following:

- `auto`: Spreading is on if the timing effort is set to high (`flow_effort` attribute to `high` in the `init` Category Attributes). Otherwise it is off.
- `true`: 0.5 / 1 track spreading. Set `route_detail_post_route_spread_wire` to `true` if you want to spread.
- `false`: no spreading
- `0.25`: 0.25 / 0.5 / 1 track spreading
- `0.5`: 0.5 / 1 track spreading
- `1`: 1 track spreading

Note: To turn off this feature, set `route_detail_post_route_spread_wire` to `false` explicitly.

To call wire spreading explicitly, set `route_detail_post_route_spread_wire` to `true` and call `route_design -wire_opt`.

Post route wire spreading automatically avoids fractional spreading for DPT designs. If `route_detail_post_route_spread_wire` is enabled in 20nm designs and below, only 1 track wire spreading is done on DPT layers.

Related Commands

- route_detail
- route_global_detail
- route_design

route_detail_post_route_swap_via

```
route_detail_post_route_swap_via {true | multicut | false | none}
```

Default: `false`

Data_type: string, read/write

Swaps single-cut vias for multiple-cut vias or reverses swapping on critical nets in a fully routed design, so that multiple-cut vias are swapped for single-cut vias on those nets. It swaps single-cut vias to multi-cut vias including vias of non-default nets. However, while swapping multi-cut vias to single cut vias, NanoRoute does not swap the vias of non-default rule nets. In timing-driven routing, does not swap vias if doing so would negatively impact timing.

A net with a pin on a critical path is considered a critical net if the pin slack, plus the margin defined by the `route_detail_min_slack_for_opt_wire` attribute, is less than or equal to 0. Use this attribute with the `route_design` command. You do not have to specify a value for this parameter when you run the following command, because the software does so automatically:

```
route_design -via_opt
```

Note: This attribute is not enabled concurrently with postroute wire widening or wire spreading. In addition, the software does not perform via swapping if `route_detail_post_route_litho_repair true` is specified.

Specify one of the following:

- `true`: Swaps all nets. The nets with attributes will get double cut and DFM precedence for others. Its weight can be specified with the `route_via_weight` attribute.
- `multicut`: All nets will be swapped with dcut precedence.
- `none`: Does not swap vias.

- `false`: Only nets with attributes are swapped.

Related Commands

- `route_detail`
- `route_global_detail`
- `route_design`

route_detail_post_route_via_pillar_effort

`route_detail_post_route_via_pillar_effort {none | low | medium | high}`

Default: low

Data_type: string, read/write

Specifies the effort levels for adding via pillars. Specify one of the following:

- `none`: Adds via pillars only during routing that are required for EM constraints.
- `low`: Adds via pillars during timing-driven global routing where they help in timing and do no impact congestion
- `medium`: Adds via pillars more aggressively during routing.
- `high`: Adds via pillars throughout routing to maximize via pillar impact on timing and area.

Related Commands

- `route_detail`
- `route_global_detail`
- `route_design`

route_detail_post_route_wire_widen

`route_detail_post_route_wire_widen {widen | shrink | none}`

Default: `none`

Data_type: string, read/write

Applies the rule specified by the `route_detail_post_route_wire_widen_rule` attribute to widen or shrink wires. Use this parameter with the `route_design` command.

Note: This attribute is not enabled concurrently with postroute via swapping.

Note: Wire widening is not performed on layer Metal1.

Specify one of the following:

- `widen`: During timing-driven routing, widens only non-critical nets; otherwise widens as many nets as possible that are covered by the nondefault rule for widening wires.
- `shrink`: During timing-driven routing, shrinks only critical nets; otherwise shrinks as many nets as possible that are covered by the nondefault rule for widening wires.
- `none`: Turns off this attribute, so the router does not widen or shrink wires.

Related Commands

- `route_detail`
- `route_global_detail`
- `route_design`

route_detail_post_route_wire_widen_rule

```
route_detail_post_route_wire_widen_rule rule_name
```

Default : "" (empty string)

Data_type: string, read/write

Specifies the non-default rule for widening wires. The rule is specified in the LEF file. Use this attribute with the `route_design` command.

Related Commands

- `route_detail`
- `route_global_detail`
- `route_design`

route_detail_postroute_via_priority

```
route_detail_postroute_via_priority {auto | allNets | criticalNetsfirst | nonCriticalNetOnly}
```

Default: `auto`

Data_type: string, read/write

Sets the net priority for postRoute via swapping based on timing.

- `auto`: Specifies the current behavior of the software, which is to detect the node and timing files to determine whether to do `criticalNetsFirst` or `nonCriticalNetOnly`. This is the default behavior.
- `allNets`: Swaps single-cut vias for multiple-cut vias on all nets.
- `criticalNetsFirst`: Specifies that swapping should be preformed first on timing critical nets and then on other nets. This option is recommended for process nodes below 32nm.
- `nonCriticalNetonly`: Specifies that swapping should not be performed on timing critical nets. This option is recommended for 32 nm and above process nodes.

Related Commands

- `route_detail`
- `route_global_detail`
- `route_design`

route_detail_search_and_repair

```
route_detail_search_and_repair {true | false}
```

Default: `true`

Data_type: bool, read/write

Runs a search-and-repair step after the initial detailed routing (end iteration 20).

Tip: Specify `0` for this attribute and run detailed routing to determine quickly if the design is routable.

Related Commands

- `route_detail`
- `route_global_detail`
- `route_design`

route_detail_signoff_effort

```
route_detail_signoff_effort {high | medium | low | auto | n}
```

Default: `high`

Data_type: string, read/write

Specifies when to skip fixing violations related to fixed objects.

- `high`: In this mode, the software will ignore violations related to fixed objects after 20 detail route iterations and will continue with post-route operations.
- `medium`: When set to `medium`, the software will ignore any violations related to fixed objects after detail route iteration number 15.
- `low`: When set to `low`, the software will ignore any violations related to fixed objects after detail route iteration number 5 and it will also skip post-route operations.
- `auto`: When set to `auto`, the software will automatically detect when to ignore violations related to fixed objects and it will also skip post-route operations.
- `n`: You can also specify the number of iterations or "n". When the iteration number is specified, the software will ignore violations related to fixed objects after the specified number of detail route iteration and will also skip post-route operations.

Note: n must be between 3 and 20.

Related Commands

- `route_detail`
- `route_global_detail`
- `route_design`

route_detail_stub_routing_in_first_layer

```
route_detail_stub_routing_in_first_layer {true | false}
```

Default: `false`

Data_type: bool, read/write

Specifies that minimum routing should be done on the first routing layer defined in the LEF.

Related Commands

- `route_detail`
- `route_global_detail`
- `route_design`

route_detail_use_multi_cut_via_effort

```
route_detail_use_multi_cut_via_effort {low | medium | high}
```

Default: `low`

Data_type: string, read/write

Specifies the effort level toward increasing the ratio of double-cut vias to single-cut vias concurrently with routing. Increasing the effort level increases the double-cut via ratio and decreases the total number of vias in the design, which means that it also decreases the number of single-cut vias. The lower the number of single-cut vias, the better the yield will be. After routing with this parameter specified, you can add more double-cut vias or larger overhang vias by using the `route_design` command.

Specify one of the following:

- `low`: Specifies normal routing, so the router uses single-cut vias only.
- `medium`: Balances the need for a high double-cut viaration with run time and congestion. Specifying this parameter increases the run time somewhat compared with the low parameter. The router inserts some double-cut vias, although not as many as if the high attribute were specified.
- `high`: Specifies the highest yield possible for vias, as the router puts its best effort toward achieving the highest possible double-cut via ratio at the expense of run time and congestion.

Related Commands

- route_detail
- route_global_detail
- route_design

route_diode_insertion_for_clock_nets

```
route_diode_insertion_for_clock_nets {true | false}
```

Default: `false`

Data_type: bool, read/write

Inserts diodes to repair process antenna violations on clock nets that are in the regular net section of the DEF file.

Related Commands

- route_detail
- route_global_detail
- route_design

route_eco_ignore_existing_route

```
route_eco_ignore_existing_route value
```

Default: " "

Data_type: string, read/write

When specified, the selected net will be routed with highest priority by ignoring existing routes in post-route mode.

- `all`: Ignores all other nets in the design that are not selected, including clock, secondary PG, and signal nets.
- `signal`: Ignores only other signal nets that are not selected, but honors clock and secondary PG nets.

For example, the following commands select and route a list of nets by ignoring only other signal nets, and considers secondary PG and clock nets during routing. After that, a second route_eco is run to fix DRC violations:

```
select_obj [get_db nets {net1 net2 net3}]
set_db route_selected_net_only true
set_db route_eco_ignore_existing_route signal
route_eco
```

```
reset_db -category route
set_db route_selected_net_only
set_db route_eco_ignore_existing_route
verify_drc -limit 0
route_eco
```

Related Commands

- route_detail
- route_global_detail
- route_design

route_enable_route_rule_si_limit_length

```
route_enable_route_rule_si_limit_length {true | false}
```

Default: `false`

Data_type: string, read/write

Use MAR wire length as NDR PRL requirement for metal spacing check.

Related Commands

- route_detail
- route_global_detail
- route_design

route_enforce_route_rule_on_special_net_wire

```
route_enforce_route_rule_on_special_net_wire {false | true | special_net_name_list}
```

Default: false

Data_type: string, read/write

Specifies that non-default rule will be enforced on all special wire segments of specified nets.

Specify one of the following parameters:

- false: When specified, the non-default rule is not enforced on any wire segment of any special net. It is the default value.
- true: When specified, the non-default rule is enforced on all wire segments of all special nets (with non default rules attached).
- specified_special_net_list: When specified, the non-default rule is enforced on all wire segments of those specified special nets.

Related Commands

- route_detail
- route_global_detail
- route_design

route_extra_via_enclosure

```
route_extra_via_enclosure distance
```

Default: 0.0

Data_type: double, read/write

Specifies an extra via enclosure to use when connecting to block pins and special wires. To center vias for via access, specify a value that is equal to one-half the width of the largest pin or wire. An extra enclosure of this size forces NanoRoute to connect tie-high and tie-low nets to the center of power routes.

Related Commands

- route_detail
- route_global_detail
- route_design

route_fix_clock_nets

```
route_fix_clock_nets {true | false}
```

Default: false

Data_type: bool, read/write

Specifies whether to maintain the current status of clock nets when the route_design command runs or change the status to ROUTED.

- If you specify true for this parameter, route_design does not change the status of clock nets.
- If you do not specify this parameter (or you specify false for this parameter), route_design changes the status of clock nets changes to ROUTED.

Note: The parameter applies to the route_design command only. Not to route_detail, or route_global_detail.

Related Commands

- route_design

route_high_freq_constraint_groups

```
route_high_freq_constraint_groups {order net match bus pair shield}
```

Default: " "

Data_type: string, read/write

Routes only the specified constraint groups. The constraint groups can be - `net match bus pair shield`.
Note: You can add "order" in front to control the route order. If order is present, the NanoRoute high frequency router will honor the order, if not, the router will determine the order.

Related Commands

- route_design

route_high_freq_match_report_file

```
route_high_freq_match_report_file filename
```

Default: " "

Data_type: string, read/write

Specifies the name of the match report file.

Related Commands

- route_design

route_high_freq_num_reserved_layers

```
route_high_freq_num_reserved_layers value
```

Default: 1

Data_type: int, read/write

Specifies the number of metal layers to reserve for high frequency routing above the standard cell area so that routing has more resources to access standard cell pins.

Related Commands

- route_design

route_high_freq_remove_floating_shield

```
route_high_freq_remove_floating_shield {true|false}
```

Default: `false`

Data_type: bool, read/write

Removes floating shield segments for high frequency nets.

Related Commands

- route_design

route_high_freq_search_repair

```
route_high_freq_search_repair {auto | false | true | only}
```

Default: `auto`

Data_type: string, read/write

Runs search and repair to remove violations.

Related Commands

- route_design

route_high_freq_shield_trim_length

```
route_high_freq_shield_trim_length value
```

Default: `0.0`

Data_type: double, read/write

Specifies the shielding wire length. All shielding wires longer than the specified value are preserved by NanoRoute for high frequency nets.

Related Commands

- route_design

route_honor_exclusive_region

```
route_honor_exclusive_region {true | false}
```

Default: `false`

Data_type: double, read/write

When set to true, it enables the NanoRoute router to honor exclusive region routing.

Related Commands

- route_detail
- route_design

route_honor_power_domain

```
route_honor_power_domain {true | false}
```

Default: `false`

Data_type: bool, read/write

Routes nets without crossing power domains.

- If a net connects pins that all belong to one power domain, the software tries to route the net within that power domain. If completing the routing is not possible without breaking the rules, due to congestion for example, completes the routing and issues a warning message.
- If a net connects pins that belong to different power domains, the software tries to route the net within the involved power domains, without crossing over into other power domains that do not include any of the pins in the net. If completing the routing is not possible without breaking this rule, completes the routing and issues a warning message.

Note: You must specify a power domain before using this attribute.

Related Commands

- route_detail
- route_global_detail

- route_design

route_ignore_antenna_top_cell_pin

```
route_ignore_antenna_top_cell_pin {true | false}
```

Default: `true`

Data_type: bool, read/write

If set to true, ignores any process antenna data from external I/O pins (e.g. DEF PIN ANTENNA* data), so only the routing and antenna data from cells inside the design are used to find and repair local antenna violations. This is most commonly used with methodologies that insert diodes at the boundary pins (either inside the block or at the next level up), or in bottom-up flows where the block is routed, updated antenna data is created with `check_process_antenna`, and then passed up to the top-level in an updated block abstract for the top-level to resolve any violations with added diodes or layer-hops.

If set to 0, the external IO antenna data is used to find and repair violations. This option will add protection diodes if needed to any external output/inout IOs that have antenna data that requires protection (e.g. have gate area), and will include the effects of external antenna data when finding and fixing local antenna violations. This is used when the user has “antenna budgets” added to the external IOs that estimate the worst-case top-level routing and gate-areas that needs protection inside this block. Currently these types of antenna budgets can only be added with defIn or oaIn.

Related Commands

- route_detail
- route_global_detail
- route_design

route_ignore_follow_pin_shapes

```
route_ignore_follow_pin_shapes {true | false}
```

Default: `false`

Data_type: bool, read/write

Specifies that follow pin via shapes will be ignored when this attribute is set to true.

Related Commands

- route_detail
- route_global_detail
- route_design

route_interposer_allow_diagonal_trunk

```
route_interposer_allow_diagonal_trunk {auto | true | false}
```

Default: `" "`

Data_type: string, read/write

Allows you to control the trunk patterns in interposer routing.

- `auto`: Allows the routing to decide the diagonal or straight trunk patterns based on the routing resources.
- `true`: Enables diagonal trunk in interposer routing.
- `false`: Disables diagonal trunk in interposer routing.

Related Commands

- route_detail
- route_global_detail
- route_design

route_interposer_interlayer_shielding_layers

`route_interposer_interlayer_shielding_layers "layer_name direction layer_name1:layer_name2 direction …"`

Default: `" "`

Data_type: string, read/write

Specifies the signal layers that require shielding along with the direction - top, bottom, both. By default, all routing layers (used by current routing command) are shielded from bottom.

For example, the following command specifies that the M2/M3/M4 signal layers will be shielded from the top, M7 from the bottom, and M9 will be shielded from both sides:

```
set_db route_interposer_interlayer_shielding_layers "M2:M4 top M7 bottom M9 both"
```

Related Commands

- route_detail
- route_global_detail
- route_design

route_interposer_interlayer_shielding_nets

`route_interposer_interlayer_shielding_nets "layer_name1:layer_nameN net_name …"`

Default: `" "`

Data_type: string, read/write

Specifies the inter-layer shielding nets.

For example, the following command specifies that the M3/M4/M5/M6 shielding net name is VSS and the M8/M9/M10 shielding net name is VDD:

```
set_db route_interposer_interlayer_shielding_nets "M3:M6 VSS M8:M10 VDD"
```

Related Commands

- route_detail
- route_global_detail
- route_design

route_interposer_interlayer_shielding_offsets

`route_interposer_interlayer_shielding_offsets "layer_name offset layer_name1:layer_name2 offset …"`

Default: `" "`

Data_type: string, read/write

Specifies the center-to-center offset values between the shielding and signal wires. By default, all routing layers (used by current routing command) have an offset value of 0.

For example, the following command specifies that the offset value is 1.1 um between the M1 shielding wire and signal wire and the offset value is 2.2 um between M2:M3 shielding wire and signal wire:

```
set_db route_interposer_interlayer_shielding_offsets "M1 1.1 M2:M3 2.2"
```

Related Commands

- route_detail
- route_global_detail
- route_design

route_interposer_interlayer_shielding_widths

`route_interposer_interlayer_shielding_widths "layer_name width layer_name1:layer_name2 width …"`

Default: " "

Data_type: string, read/write

Specifies the inter-layer shielding widths. By default, all routing layers (used by current routing command) are shielded by the default width wires.

For example, the following command specifies that the shielding M1 wires width is 1.1 um, shielding M2/M3 wires width is 2.2 um:

```
set_db route_interposer_interlayer_shielding_widths "M1 1.1 M2:M3 2.2"
```

Related Commands

- route_detail
- route_global_detail
- route_design

route_interposer_same_layer_shielding_net

```
route_interposer_same_layer_shielding_net{net_name}
```

Default: " "

Data_type: string, read/write

Specifies the same-layer shielding nets.

For example, the following commands specifies that the M1/M3/M5 shielding net name is VSS:

```
set_db route_interposer_trunk_routing_layers {M1 M3 M5}
set_db route_interposer_same_layer_shielding_net {VSS}
```

Related Commands

- route_detail
- route_global_detail
- route_design

route_interposer_same_layer_shielding_width_spacing

```
route_interposer_same_layer_shielding_width_spacing {width width}
```

Default: " "

Data_type: string, read/write

Specifies the same-layer shielding widths.

For example, the following commands specify that the shielding width space between M1/M3/M5 is 0.6 and 1.2:

```
set_db route_interposer_same_layer_shielding_width_spacing {0.6 1.2}
```

Related Commands

- route_detail
- route_global_detail
- route_design

route_interposer_trunk_routing_layers

```
route_interposer_trunk_routing_layers {layers}
```

Default: " "

Data_type: string, read/write

Specifies the trunk routing layers.

For examples, the following command sets layers M1, M3, and M5 as the interposer trunk routing layers.

```
set_db route_interposer_trunk_routing_layers {M1 M3 M5}
```

Related Commands

- route_detail
- route_global_detail
- route_design

route_interposer_trunk_routing_width_spacing

```
route_interposer_trunk_routing_width_spacing {width width}
```

Default: " "

Data_type: string, read/write

Specifies the trunk routing width spacing.

For example,

```
set_db route_interposer_trunk_routing_width_spacing {1.0 1.0}
```

Related Commands

- route_detail
- route_global_detail
- route_design

route_number_fail_limit

```
route_number_fail_limit integer
```

Default: 0

Data_type: int, read/write

Specifies the maximum number of command fails allowed. In interactive mode, if a command fails you can continue with the next command. But in batch mode, if one command fails, the script stops at the failed command, however you can increase the fail limit to let it continue.

Related Commands

- route_detail
- route_global_detail
- route_design

route_number_thread

```
route_number_thread numberProcessors
```

Default: 1

Data_type: int, read/write

Note: This attribute is included for backward compatibility. Cadence recommends that you use the `set_multi_cpu_usage` command to specify the number of threads in multiple-CPU processing.

Specifies the maximum number of threads to use in multi-threading. Generally, using more threads decreases the run time. Specify this attribute before running any routing commands. There is no absolute limit to the number of threads the router can use, however, Cadence recommends that you do not specify more threads than the number of CPUs available. The router does not necessarily use the number of threads you specify – it determines the optimal number up to the value you specify. A small design generally uses fewer threads than a large design.

Note: To enable a value greater than 1 for this attribute, you must have licenses available for the additional threads.

Related Commands

- route_detail

- route_global_detail
- route_design

route_number_warning_limit

route_number_warning_limit *integer*

Default: 0

Data_type: int, read/write

Specifies the maximum number of warning messages for any one command reported to the log file. After reaching this number, the router continues processing, but does not report warnings for this command.

Related Commands

- route_detail
- route_global_detail
- route_design

route_process_node

route_process_node *value*

Default: " "

Data_type: string, read/write

Specifies the process node. This attribute is an umbrella option to set the process related settings for routing. It supports the following values:

- S2: Sets the Samsung 2nm process related settings.
- N2: Sets the TSMC 2nm process related settings.
- N2X: Sets the TSMC N2X process related settings.
- S3: Sets the Samsung 3nm process related settings.
- N3: Sets the TSMC 3nm process related settings.
- N3E: Sets the TSMC N3E process related settings.
- S4 : Sets the Samsung 4nm process related settings.
- I5 : Sets the INTEL 5nm process related settings.
- N6: Sets the TSMC 6nm process related settings
- C7: Sets the SMIC 7nm process related settings.
- G7: Sets the GlobalFoundries 7nm process related settings.
- I7 : Sets the INTEL 7nm process related settings.
- N7: Sets the TSMC 7nm process related settings.
- S7: Sets the Samsung 7nm process related settings.
- I10: Sets the ICF 10nm process related settings.
- N10: Sets the TSMC 10nm process related settings.
- SU10: Sets the Samsung 10LPU process related settings.
- S11: Sets the Samsung 11nm process related settings.
- G12: Sets the GlobalFoundries 12nm process related settings.
- N12: Sets the TSMC 12nm process related settings.
- G14: Sets the GlobalFoundries 14nm process related settings.
- N22: Sets the TSMC 22nm process related settings.

Related Commands

- route_detail
- route_global_detail
- route_design

route_rc_extraction_corner

```
route_rc_extraction_corner value
```

Default: " "

Data_type: string, read/write

Specifies the RC extraction corner to use in routing. This helps in better resistance calculation.

Related Commands

- route_detail
- route_global_detail
- route_design

route_relaxed_route_rule_spacing_to_power_ground_nets

```
route_relaxed_route_rule_spacing_to_power_ground_nets {layer:spacing layer:spacing}
```

Default: none

Data_type: string, read/write

Relaxes the spacing requirement from NDR spacing for the layers. *Layer_spacing* signifies a distance in microns (per layer) of how close any net with NDR can get to to the power/ground stripes.

For example,

```
set_db route_strict_honor_route_rule true
set_db route_relaxed_route_rule_spacing_to_power_ground_nets {metal1:0.04 metal2:0.04 metal3:0.04 metal4:0.04 metal5:0.04}
```

It only works for PG usage net and relaxes the NDR requirement to the specified spacing. The specified spacing should be greater than the `min_spacing` and less than the `hard_ndr_spacing`.

Related Commands

- route_detail
- route_global_detail
- route_design

route_reserve_space_for_multi_cut

```
route_reserve_space_for_multi_cut {true | false}
```

Default: `false`

Data_type: bool, read/write

Reserves space to insert multicut vias in postroute stage. This option has to be set before routing. After routing with this parameter specified, you can add double-cut vias or larger overhang vias by using the `route_design -via_opt` command.

Note: `route_reserve_space_for_multi_cut` is `true` by default for 32nm designs and below.

Related Commands

- route_detail

- route_global_detail
- route_design

route_reverse_direction

```
route_reverse_direction "(x1 y1 x2 y2) (x3 y3 x4 y4) ...."
```

Default : {}

Data_type: string, read/write

Reverses the preferred routing direction in the specified areas. You can specify one or more areas.

Note: To route a specified net in the non preferred direction in the specified area, you must also use the following commands:

```
select_obj [get_db nets net1]
route_selected_net_only 1
```

You can also use this parameter to increase the utilization of routing channels where some routing layers are blocked by power routing. For example, in a narrow routing channel, the long way usually has more traffic than the crosswise direction. If the preferred direction for power routing is the long way, the power routing might use too many routing resources and cause congestion. Reversing the preferred routing direction in this channel, that is, using routing layers without power wires for long way routing, might increase available routing resources.

Related Commands

- route_detail
- route_global_detail
- route_design

route_route_clock_nets_first

```
route_route_clock_nets_first  {true | false}
```

Default: true

Data_type: bool, read/write

Detects whether clock nets require ECO routing after an route_design super command checks whether the existing clock nets were fully or partially routed with the NanoRoute router and reroutes them before routing the remaining nets. If the existing clock nets were not routed with NanoRoute, route_design skips routing on those nets.

Related Commands

- route_detail
- route_global_detail
- route_design

route_selected_net_only

```
route_selected_net_only {true | false}
```

Default: false

Data_type: bool, read/write

Specifies whether NanoRoute routes all nets at once or routes selected nets only. To route critical nets as short as possible, select the critical nets and set this attribute to true. NanoRoute does the following:

- Removes incomplete nets unless they are marked + FIXED or have a -skip_routing net attribute.
- Routes the remaining selected nets.

Note: To select a net, use the select_obj command or click the net in the design display window.

Related Commands

- route_detail
- route_global_detail
- route_design

route_shield_crosstie_offset

```
route_shield_crosstie_offset {layerName:numTrack1 layerName2:numTrack2...}
```

Default: " "

Data_type: string, read/write

Specifies the offset in terms of number of tracks for adding crossties. It enables you to control the starting track for adding crossties. Default is 0 offset for all layers.

Related Commands

- route_detail
- route_global_detail
- route_design

route_shield_report_skip_status

```
route_shield_report_skip_status {true|false}
```

Default: `false`

Data_type: bool, read/write

Generates a report file containing a list of nets that have been skipped for shielding.

Related Commands

- route_detail
- route_global_detail
- route_design

route_shield_tap_cell_insertion

```
route_shield_tap_cell_insertion {true | false}
```

Default: `false`

Data_type: `bool`

When set to true, it enables the NanoRoute router to insert tap cells for shielding.

Related Commands

- route_detail
- route_global_detail
- route_design

route_shield_tap_cell_name

```
route_shield_tap_cell_name string
```

Default: " "

Data_type: string

Specifies the tap cell names for shielding.

Related Commands

- route_detail
- route_global_detail
- route_design

route_skip_analog

```
route_skip_analog {true | false}
```

Default: `false`

Data_type: bool, read/write

Skips routing nets or pins marked `+ USE ANALOG` in the DEF file or where `signalType` modified to `analog` in the OA DB. Specify this parameter for NanoRoute to skip the routing of nets with Analog Signal Type.

Related Commands

- route_detail
- route_global_detail
- route_design

route_strictly_honor_1d_routing

```
route_strictly_honor_1d_routing {true | false | bottomLayerNum:topLayerNum}
```

Default: `false`

Data_type: string, read/write

When specified, the NanoRoute router strictly honors 1d routing. This enables you to turn off jog routing on a per layer basis.

Related Commands

- route_detail
- route_global_detail
- route_design

route_strict_honor_route_rule

```
route_strict_honor_route_rule {true | false | bottom_layer_num:top_layer_num | wire bottom_layer_num:top_layer_num}
```

Default: `false`

Data_type: string, read/write

Uses nondefault spacing rules for nondefault wires.

Note: The router tapers wires and uses default vias even when this parameter is set to `true`.

You define nondefault routing rules in the `NONDEFAULTRULE` statement of the LEF file. Specify one of the following arguments:

- `false`: Specifies that the nondefault rule (NDR) spacing is a soft rule. In congested areas NanoRoute may need to use the extra NDR spacing to meet the minimum foundry rule set.
- `true`: Specifies that the nondefault rule spacing is a hard rule. NanoRoute will attempt to honor the requested NDR spacing irrespective of resource availability.
- *bottom_layer_num*:*top_layer_num*: Specifies that the nondefault rule spacing is a hard rule for all the layers specified in the layer range.

- `wire bottom_layer_num:top_layer_num`: Specifies that the nondefault rule spacing is a hard rule for the wire but not the via metal to enable better DFM via optimization.

For example,

- `set_db route_strict_honor_route_rule 3:6`
  Specifies that the NDR spacing will be used in layers Metal3, Metal4, Metal5, and Metal6 for wire and via metal.
- `set_db route_strict_honor_route_rule "wire 3:6"`
  Specifies that the NDR spacing will be used in Metal3, Metal4, Metal5, and Metal6 for wire and not via metal.

Note: Setting the rule to `false` will attempt to honor the NDR rule and allow short hot spots without detouring. If you set the rule to `true` (hard rule) we recommend you work more on minimizing the congested areas to preserve the NDR topology.

Related Commands

- `route_detail`
- `route_global_detail`
- `route_design`

route_stripe_layer_range

```
route_stripe_layer_range "bottomLayerNum:topLayerNum"
```

Default : `{}`

Data_type: string, read/write

Specifies the range of routing layers for stripes.

Related Commands

- `route_detail`
- `route_global_detail`
- `route_design`

route_third_party_data

```
route_third_party_data {true | false}
```

Default : `false`

Data_type: bool, read/write

When enabled, NanoRoute supports third party data as input. It accepts non-DEF connectivity models, multiple NDRs per net, and other third party structures in order to enable ECO routing on non-native data.

Note: Since the third party data uses DEF data types for different purposes and is formed for a completely different use model, it is important to translate it into Cadence types. With proper setup Innovus accepts third party data for ECO Routing.

Related Commands

- `route_detail`
- `route_global_detail`
- `route_design`

route_tieoff_to_shapes

```
route_tieoff_to_shapes "auto | stripe | ring | followpin | powergroundpin"
```

Default: `auto`

Data_type: string, read/write

Specifies the target to special wire shapes or target instance pin for tie net connection.

Specify one of the following arguments:

- `auto`: Specify `auto` to target special wire shapes, including stripe, ring, and standard cell followpin.
- `stripe`: Specify `stripe` to target stripes only. Stripes are the special wiring used for creating the power mesh.
- `ring`: Specify `ring` to target only rings. The power mesh along the macro boundary forms a ring. If it is in a macro placed in the design, only exposes power ring-pin shapes. When power stripes are added across the macro, they connect to each side of the ring-pin but do not go inside the ring.
- `followpin`: Specify `followpin` to target standard cell followpin only. Followpin is the special wiring used for connecting standard cells to power mesh.
- `powergroundpin`: Specify `powergroundpin` to target the power and ground pin of the instances only.
- `auto powergroundpin`: Specifies the target to special wire shapes or target instance pin for tie net.

Note: The target shapes can be specified together. The value `auto` can only be specified with `powergroundpin`, since the other values stripe, ring, and followpin are included in auto.

For example,

- `route_tieoff_to_shapes "followpin stripe"` signifies that it can tie to followpin or stripe.
- `route_tieoff_to_shapes "followpin powergroundpin"` signifies that it can tie to followpin or power ground pin
- `route_tieoff_to_shapes "stripe powergroundpin"` signifies that it can tie to stripe or power ground pin

Note: Using "`auto stripe`" is not recommended as `stripe` is included by auto.

Related Commands

- `route_detail`
- `route_global_detail`
- `route_design`

route_trim_pull_back_distance_from_boundary

`route_trim_pull_back_distance_from_boundary {`*`layer`*`:`*`value ...`*`}`

Default: " "

Data_type: string, read/write

Specifies the boundary pullback value per layer for trim metal layers. The pullback value signifies how far the trim metal router brings signals and fill metal to a boundary (for example, block edge) in the absence of any obstructions.

Related Commands

- `route_detail`
- `route_global_detail`
- `route_design`

route_trunk_with_cluster_target_size

`route_trunk_with_cluster_target_size` *`integer`*

Default: `1`

Data_type: int, read/write

Controls the routing pattern of nets with the trunk pattern attribute. By default, the router connects each pin in nets on which you have applied `set_attribute -pattern trunk` directly to the trunk of the special net; use this parameter to change the routing pattern for those nets:

- Specify 0 to route nets with the trunk pattern specified with a Steiner tree pattern instead of a trunk pattern.
- Specify an integer greater than 1 to route the nets with a fishbone pattern. In a fishbone pattern, several pins in a column are clustered together before connecting to the special net. The value of this attribute represents the approximate number of pins in each cluster.

Related Commands

- route_detail
- route_global_detail
- route_design

route_use_auto_via

`route_use_auto_via *value*`

Default: `auto`

Data_type: string, read/write

When specified, it allows the use of internal generated vias. In new advance nodes, it enables you to specify some custom via's names from the LEF for the router to use. This helps to provide better pin access. The following is the syntax of the supported values:

`{true | false| auto | {-rule *ruleName1 viaName viaName*} {-rule *ruleName2 viaName viaName*} {-prefix_name *viaPrefix* }}`

Specify one of the following arguments:

- `true`: Specify `true` to turn on auto via generation.
- `false`: Specify `false` to turn off auto via generation.
- `auto`: Specify `auto` to enable via generation based on process nodes and allow Innovus to decide to turn on or off.
- `-rule *ruleName1 viaName viaName*`: Specify `-rule` to enable Innovus to use the provided via list for the specified rule.
- `-prefix_name`: Specify `-prefix_name` to add a prefix to the auto generated via names. By default, `NR_` prefix is used by the tool.

For example,

- `set_db route_use_auto_via "-rule default VIA0_1 VIA0_2"`
  signifies that M0 pins use VIA0_1 and VIA0_2 for default routing.
- `set_db route_use_auto_via {-rule HV_NDR HV_NDR_VIA4_2x1 HV_NDR_VIA5_2x1 HV_NDR_VIA6_2x1HV_NDR_VIA7_2x1}`
  signifies that for nets with HV_NDR rule, use the specified list via's only for those layers. For other layers Innovus can choose other via's defined in the NDR rule section or it can auto generate. Only 5 via's per layer per rule can be defined.
- **Note**: The NanoRoute router treats VIA and USEVIA definitions as soft constraints when the `route_use_auto_via attribute` is enabled (default is enabled). The vias specified with VIA or USEVIA LEF rules are considered as target vias along with other auto-generated vias. To specify a hard constraint for the vias to use, use the USEVIACUTCLASS rule. Alternatively, specify the following:
  `set_db route_use_auto_via "-rule *ndr_rule_name viaName*"`
- `set_db route_use_auto_via "-prefix_name blockA"`
  Signifies that the NanoRoute router will use blockA as the prefix for auto generated vias names instead of NR_.

Related Commands

- route_detail
- route_global_detail
- route_design

route_via_weight

`route_via_weight {*viaName viaWeight*}`

Default : `{}`

Data_type: string, read/write

Specifies the priority or the order of vias for post route via swap to use. It sets the via weight for the specified via. The bigger the number, the higher priority the via is given. A negative value means the via will not be used.

Range: `-1 to 256`

You can use wildcards for the *viaName*.

For example:

```
route_via_weight {*FBD* 8, *FBS* 7, *PBD* 6, *PBS* 5, *2cut_p1* 4, *2cut_p2* 3, *2cut_p3* 2, *FAT* 1}
```

Note: Via name can be a predefined class of vias such as @1cut and @2cut. Here, @1cut stands for single cut vias, @2cut stands for double cut vias.

In case of duplicated value setting of via, the latest value will be used.
For example:

```
route_via_weight {@1cut 2 @2cut 4 VIA12_1cut 1}
```

Here, VIA12_1cut was first set weight to 2, and then set to 1, the final weight of VIA12_1cut is 1.

When you specify this option to do post route via swap, it's important to set the related options to improve the DFM and double cut via ratio.

- Specify `route_reserve_space_for_multi_cut 1` before detail route to reserve space for inserting multicut vias in postroute via swap stage
- Specify `route_with_timing_driven 0` and `route_detail_post_route_swap_via multiCut` to swap the single cut via to DFM and double cut via as much as possible

Related Commands

- route_detail
- route_global_detail
- route_design

route_with_eco

```
route_with_eco {true | false}
```

Default: `false`

Data_type: bool, read/write

Routes in engineering change order (ECO) mode if route_global_detail is specified. During ECO mode, NanoRoute completes partial routes with added logic, while maintaining the existing wire segments as much as possible.

Note: The Innovus software also has an `route_eco` command.

Related Commands

- route_detail
- route_global_detail
- route_design

route_with_litho_driven

```
route_with_litho_driven {true | false}
```

Default: `false`

Data_type: bool, read/write

Avoids lithography problems during routing by avoiding certain routing patterns that might lead to the creation of lithography hotspots.

Related Commands

- route_detail
- route_global_detail
- route_design

route_with_si_driven

```
route_with_si_driven {true | false}
```

Default: `false`

Data_type: bool, read/write

Prevents or reduces crosstalk. Works in conjunction with timing-driven routing.

- When timing-driven routing is specified, uses SMART routing to identify victim nets and minimize crosstalk by wire spacing, layer hopping, net ordering, and minimizing the use of long parallel wires.
- When timing-driven routing is not specified, uses an older signal integrity engine to minimize crosstalk by preventing or reducing the use of long parallel wires.

Specify `route_with_timing_driven` and `route_with_si_driven` to run SMART routing.

Related Commands

- `route_detail`
- `route_global_detail`
- `route_design`

route_with_timing_driven

```
route_with_timing_driven {true | false}
```

Default: `false`

Data_type: bool, read/write

Minimizes timing violations by analyzing the timing slack for each path, the drive strengths of each cell in the library, and the maximum capacitance and maximum transition limits. During timing-driven routing, NanoRoute routes multi-pin nets to the most critical sink first, performs wire optimization by reducing resistance and coupling, and continually adjusts detouring.

When this attribute is set to `1`, the router automatically generates a timing information file named `.timingfile.tif` in the working directory. It uses this file when it runs timing driven routing.

Note: Specify the following commands and to skip the CTE call and timing file generation step and use the specified file:

```
write_design_timing  fileName
set_db route_with_timing_driven 1
```

Specify `route_with_timing_driven` and `route_with_si_driven` to run SMART routing.

You can use the `route_with_timing_driven` attribute to reduce the timing closure effort during post-route ECO. NanoRoute will replace via to multi-cut via on non-critical nets, if a violation does not occur.

Related Commands

- `route_detail`
- `route_global_detail`
- `route_design`

route_with_trim_metal

```
route_with_trim_metal value
```

Default: `" "`

Data_type: string, read/write

Specifies the preferred trim metal grid for NanoRoute Router to use. You can specify the *value* in the following format:

```
{-layer layer_number -mask1 {-pitch value -core_offset value} -mask1 { -pitch value ... } -mask2 { -pitch value ... } -mask2 {-pitch value ... }}
```

For example, the following command specifies the preferred trim metal grid on layer 2.

```
set_db route_with_trim_metal {-layer 2 -mask1 {-pitch 0.300 -core_offset 0.025 } \
-mask1 {-pitch 0.300 -core_offset 0.195 } \
-mask2 {-pitch 0.300 -core_offset 0.065 } \
-mask2 {-pitch 0.300 -core_offset 0.235  }}
```

Note: The width value is taken from the LEF.

Related Commands

- route_detail
- route_global_detail
- route_design

route_with_via_in_pin

```
route_with_via_in_pin {true | false | bottomLayerNum:topLayerNum}
```

Default: false

Data_type: string, read/write

Encloses via geometries completely inside standard cell pins. Specify true for this parameter to avoid MINSTEP violations for standard cell pin access.

Note: To avoid violations, specify the same value for this parameter throughout the entire design flow.

- true: Enclose vias inside standard cell pins on all layers.
- false: The router is not required to enclose vias inside standard cell pins.
- *bottomLayerNum:topLayerNum:* Enclose vias inside standard cell pins on layers in the specified layer range.
  - 1:1: Specifies that Metal1 pins must enclose vias.
  - 2:5: Specifies that pins on Metal2, Metal3, Metal4, and Metal5 must enclose vias. Pins on other layers are not required to enclose vias.

Related Commands

- route_detail
- route_global_detail
- route_design

route_with_via_only_for_block_cell_pin

```
route_with_via_only_for_block_cell_pin {true|false}
```

Default: false

Data_type: string, read/write

Encloses via in pin shape for macro cell pins.

Related Commands

- route_detail
- route_global_detail
- route_design

route_with_via_only_for_stdcell_pin

```
route_with_via_only_for_stdcell_pin {true | false |  bottomLayerNum:topLayerNum }
```

Default: false

Data_type: string, read/write

Allows standard cell pins to be accessed by vias only. Prohibits planar access.

- true: Always use vias for standard cell pin access on all layers.
- false: Use vias or planar access for standard cell pins on all layers.
- *bottomLayerNum:topLayerNum:* Always use vias for pin access in the specified layer range.
  - 1:1: Always use vias for access to Metal1 standard cell pins. Metal2 and above can use vias or planar access.

- `2:5`: Always use vias for access to standard cell pins on Metal2, Metal3, Metal4, and Metal5. Metal1 and Metal6 (and above) can use vias or planar access.

Note: Set this option before any routing or verification because it affects the pin access preparation steps. In addition, after you set this option, the access remains for same the entire Innovus session. Do not change the setting, even for future routing sessions that load saved data from the database or a saved DEF file.

For example, if a design with pins on Metal1 and Metal2 is routed with this option set to 1:1, the router uses vias for access to all standard cell pins on Metal1, and vias or planar access to standard cell pins on Metal2. If you save the design after routing, then import the design in a different Innovus session and route it again with this option set to true (that is, forcing the router to use vias to access all standard cell pins on all layers), you might have planar connections (from the previous routing session) to pins on Metal2. All the planar connections to those pins become violations, because they are not allowed when this option is set to true.

Related Commands

- `route_detail`
- `route_global_detail`
- `route_design`

route_trial_max_print_ignored_pad_nets

`route_trial_max_print_ignored_pad_nets` *integer*

Default: `0`

Data_type: int, read/write

Number of nets to be printed connecting a pad term to a fterm without geometry.

Related Commands

- `route_detail`
- `route_global_detail`
- `route_design`

set_max_route_layer

```
set_max_route_layer
[-help]
layer_number
```

Specifies a routing layer limit for Early Global Route, the NanoRoute router, the mixed signal router, pin assignment, standard cell placement, and clock tree synthesis (CTS). Early Global Route, the NanoRoute router, the mixed signal router, pin assignment, standard cell placement, and CTS can each specify individual "local" maximum routing layer limits.

Note: While setting the routing layer limits, when `set_max_route_layer` is called after one of the local commands, all of the local commands are called and effectively reset to the new level. However, if the local routing command is used to set a local routing limit after `set_max_route_layer` has already been used, the local routing limit supersedes the `set_max_route_layer` limit.

To print the current maximum routing layer value set for a specific feature, use one of the following category attributes:

- place Category Attributes
- route Category Attributes
- assign_pins Category Attributes

You can use the `set_max_route_layer` command at any time during the Innovus session.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `set_max_route_layer` parameter. For a detailed description of the command and all of its parameters, use the man command: `man set_max_route_layer`.
`layer_number`	Limits routing to layers at and below the specified layer. *Data_type*: int, required

Examples

- The following command sets the routing layer limit to layer *metal4*. This limits routing to layer *metal4* and below.
  ```
  set_max_route_layer 4
  ```

Related Information

- `route_design`
- Using Early Global Route for Congestion and Timing Analysis chapter in the *Innovus Stylus Common UI User Guide*
- Using the NanoRoute Router chapter in the *Innovus Stylus Common UI User Guide*

set_route_attributes

```
set_route_attributes
[-help]
[-max_fanout integer]
[-reset]
[-stripe_layer_range min_layer:max_layer]
{-nets netName  | -cell_pins cell_name : pin_name}
[-sub_net  sub_net_name  [-comp_pins comp:pin_name list ]
 [-inst_pins instance:pin_name list]]
[-weight integer]
[-route_rule  rule_name  ]
[-route_rule_effort {auto | soft | hard | no_taper}]
[-shield_nets  special_net_name  ]
[-shield_side {one_side | two_sides}]
[[[-em_route_rule ruleName] [-em_route_rule_distance double]] |
  [-em_width_rule layer width:dist pairs]]
[-hard_spacing_length_check { layer:distance  list}]
[-coaxial_shielded_layers  layer_number : layer_number ]
[-coaxial_shield_limit layer_number:layer_number]
[-coaxial_shield_interval  tracks_number |  metal_layer_name : tracks_number … ]
[-pattern {steiner|trunk}]
[-top_preferred_routing_layer layer_number]
[-bottom_preferred_routing_layer layer_number]
[-top_preferred_shielding_layer layer_number]
[-bottom_preferred_shielding_layer layer_number]
[-preferred_routing_layer_effort {low|medium|high|hard}]
[-preferred_extra_space_tracks integer]
[-avoid_detour {true|false}]
[-skip_routing {true|false}]
[-skip_antenna_fix {true|false}]
[-multi_cut_via_effort {low|medium|high}]
[-si_post_route_fix {true|false}]
[-min_stack_layer layer_number]
[-stack_distance distance]
[-mask mask_number]
[-layer_mask  bottomLayerNum:topLayerNum]
[-one_side_spacing  bottomLayerNum:topLayerNum ]
```

Attaches attributes to nets and subnets. Attaching the attributes allows the NanoRoute routing commands (`route_detail`, `route_global_detail`, `route_design`, and route Category Attributes), and the `route_early_global` command to route the nets following specific requirements. Attributes are persistent; that is, throughout the routing process, from global routing to optimization, the routers honor the attributes. The attributes are saved in the Innovus database. If you save the database and exit, the attributes remain attached to the nets when you re-import the database.

Note: Attributes are not saved in the DEF file.

To see the current attribute settings, use the `get_route_attributes` command. If you attempt to run the `set_route_attributes` command on any net that is part of a diffPair, matchLength, or complex shield constraint group, or a simple or complex netClass, the command reports an error stating that the command cannot be run on nets with integration constraints.

Tip: To revert to the default value of an attribute, specify default for the attribute.

Parameters

`-avoid_detour {true | false}`	Avoids detours of roughly more than a few gcell grids on the specified nets. This attribute affects global routing only. **Note:** Cadence recommends that you use caution with this attribute, as it adds congestion to the design. *Data_type*: string, optional
`-bottom_preferred_routing_layer layer_number`	

	Specifies the preferred lowest routing layer. This attribute is a soft limit; that is, NanoRoute might use a layer below the specified layer if necessary to complete routing. *Range:* 0-15 **Note:** Layers can only be specified using their layer ID numbers. Use of LEF layer names is not supported by this parameter. **Note:** If the top routing layer is equal to or greater than 4 and no bottom preferred routing layer is specified for clock nets, the router automatically sets the bottom preferred routing layer for clock nets to 3, to prevent the clock nets from blocking other nets. *Data_type*: string, optional
`-bottom_preferred_shielding_layer` *layer_number*	
	Specifies the preferred lowest shielding layer. **Note:** Layers can only be specified using their layer ID numbers. Use of LEF layer names is not supported by this parameter. *Data_type*: string, optional
`-cell_pins` *cell_name* : *pin_name*	Specifies attributes for all the nets connected to pins of instances having the same master cells. You can use : or / as a delimiter when you specify the cell name and the corresponding pin name. **Note:** If the specified `set_route_attributes` commands conflict, then the last setting takes precedence over all the previous `set_route_attributes` settings. *Data_type*: string, optional
`-coaxial_shield_interval` *tracks_number* | *metal_layer_name*:*tracks_number* …	
	Specifies the interval of crossties by routing track. *Range:* 0-63 *Default*: 0. Every track could be used by crosstie. For example, when `-coaxial_shield_interval 2` is specified, two tracks between two crossties will be empty. *Data_type*: string, optional
`-coaxial_shield_limit` *layer_number*:*layer_number*	
	Specifies the bottom and top routing layer limits for coaxial shields. *Data_type*: string, optional
`-coaxial_shielded_layers` *layer_number*:*layer_number*	
	Specifies the bottom and top routing layer for coaxial shields. *Data_type*: string, optional
`-comp_pins` *comp*:*pin_name list*	Specifies attributes for all nets and sub-nets connected to pins of instances having the same component. You can specify a list of pins. You can use **:** or / as a delimiter when you specify the component names and the corresponding pin names. *Data_type*: string, optional
`-em_route_rule_distance` *double*	Specifies the distance from source to sink where it is needed to be applied with the EM ndr rule. *Default*: 0 *Data_type*: string, optional
`-em_route_rule` *ruleName*	Specifies the non-default rule name which is used for EM issue fixing. *Default*: " " empty. *Data_type*: string, optional
`-em_width_rule` *layer width:dist pairs*	

	Specifies the table of layer width:distance pairs. The layer width pair denotes the width for any layer to be changed from default width for EM fixing in distance X from driver. For example, `Metal1 1.0:0.5, Metal2 2.0:1.0` **Note**: The `-em_width_rule` parameter must be used with the `-em_route_rule_distance` parameter. It can not be used with the `-em_route_rule` parameter as only one parameter can specify the non default width to apply. *Data_type*: string, optional
`-hard_spacing_length_check {`*`layer:distance`* `list}`	

	Specifies the table of parallel run length on each layer. For example, Metal1 1.0, Metal2 2.0. It relaxes a hard NDR into a soft NDR whenever the parallel run length between the NDR wire and an other wire is smaller than the specified value
	For example,
	`set_db route_strict_honor_route_rule true`
	`set_route_attributes -nets {xgxs_tx_rdy_lane[1]} -hard_spacing_length_check {metal3:0.054}`
	Where, xgxs_tx_rdy_lane[1] is the net name, metal3:0.054 is the prl limit 0.054 in M3 layer.
	This setting is for, if parallel run length is less than distance(0.054) in the layer(Metal3) for the specified net, no NDR spacing is needed for this net (wire segment).
	Data_type: string, optional

`-help`	Outputs a brief description that includes the type and default information for each `set_route_attributes` parameter. For a detailed description of the command and all of its parameters, use the man command `man set_route_attributes`
`-inst_pins` *`instance:pin_name list`*	Specifies attributes for all nets and sub-nets connected to pins of the same instance. You can specify a list of pins. You can use **:** or / as a delimiter when you specify the pin names and the corresponding instance names. *Data_type*: string, optional
`-layer_mask` *`bottomLayerNum:topLayerNum`*	
	Specifies the layer range to which the mask constraint should be applied. For example, if you specify 3:4, it means that the constraint should be applied on layers, Metal3 and Metal4. *Data_type*: string, optional
`-mask` *`mask_number`*	Masks constraint setting for the net. Legal values are from 0 to the mask number for the layers. A value of 0 indicates that there is no mask constraint on the net. A non-zero value indicates that all wiring on the net should be created on the specified mask. Layers that do not support the specified value will be treated as unconstrained. *Data_type*: string, optional
`-max_fanout` *`integer`*	Specifies the maximum fanout value for nets and subnets. Specifying `setAttribute -net` *`netname`* `-max_fanout` *`value`* provides per net based max_fanout constraint/control for a net. ◦ If `-max_fanout value` of a net is 0, and the `-pattern` of the net is `trunk`, the NanoRoute router honors the value of `get_db route_design_trunk_with_cluster_target_size` as the max fanout. ◦ If `-max_fanout value` of a net is larger than 0, and the `-pattern` of the net is `trunk`, the `-max_fanuout` value overrides the value of `get_db route_design_trunk_with_cluster_target_size` for the net. *Default*: 0 *Data_type*: string, optional
`-min_stack_layer` *`layer_number`*	Specifies the minimum stacking layer. It indicates that stacked vias should be added to the specified layer on output pin only. If a pin already has pre-defined custom stacked vias, and the upper most layer of the vias is still below the specified layer, additional stack vias should be added to the specified layer. Then, regular routing would start from the specified layer of the stacked vias. *Data_type*: string, optional
`-multi_cut_via_effort {low|medium|high}`	

	Specifies the multi-cut via effort. The effort level can be low, medium, or high. *Data_type*: string, optional
`-nets` *netName*	Specifies the net for which to set attributes. Do not enter more than one net name. • To specify more than one net, use `*` and `?` wildcard characters. • To specify non-default rule nets, use `@RULE`. • To specify clock nets, use `@CLOCK`. (To select clock nets use `selectNet -allDefClock`.) • To specify prerouted nets, use `@PREROUTED`. **Note**: Remove an attribute from a net by specifying none. For example, to remove the - shield_net attribute from net_1, specify the following command: `set_route_attributes -nets net_1 -shield_net none` *Data_type*: string, optional
`-one_side_spacing` *bottomLayerNum:topLayerNum*	
	Specifies the layer range that the one_side_spacing constraint should be applied on. This constraint means that spacing is reserved on any one side. For example, 3:4 means that the constraint should be applied on layers, Metal3 and Metal4. *Data_type*: string, optional
`-pattern {steiner | trunk}`	Specifies the routing pattern. Select one of the following: • `steiner`: Routes all pins with a single steiner tree. • `trunk`: Routes pins to the nearest special net of the same name. Creates multiple steiner trees and connects them to the trunk. *Data_type*: string, optional
`-preferred_extra_space_tracks` *integer*	
	Specifies the preferred extra spacing for nets. Gives additional pitch spacing to the specified net. Use this attribute to give critical nets extra space to reduce coupling. Specify `-preferred_extra_space_tracks 1` to give a net 1 extra pitch spacing, compared to other nets. This parameter creates a soft rule and, if the design is congested, might not give the extra space to the net. *Range:* 0-3 **Note:** If you specify this attribute, Early Global Route also gives additional space to the specified nets. *Data_type*: string, optional
`-preferred_routing_layer_effort {low | medium | high | hard}`	
	Determines how flexible the NanoRoute router is when setting layer limits for routing specified nets. Use this attribute with the `-top_preferred_routing_layer` and `-bottom_preferred_routing_layer` attributes. Specify medium, high, or hard to make NanoRoute less flexible (that is, to make NanoRoute obey the preferred routing layer range more strictly). When set to hard, NanoRoute prevents the nets from going to the non-preferred routing layers by setting large costs. *Default:* low (more flexible) *Data_type*: string, optional
`-route_rule` *rule_name*	Identifies the non-default routing rule to use with the specified net. Use this attribute for critical nets, such as clock nets, that might need to have wider wires or wider spacing than other nets. The NanoRoute router treats nondefault rule spacing as a soft attribute; that is, when routing resources are available, it honors the nondefault rule. If the area is too congested, and resources are not available, the router might not honor nondefault spacing rules. You can override this behavior by specifying `set_db route_strict_honor_route_rule true`. When you specify this parameter, the router uses the nondefault spacing rules. You define nondefault routing rules in the NONDEFAULTRULE statement of the LEF file. *Data_type*: string, optional
`-route_rule_effort {auto | soft | hard | no_taper}`	

	Specifies the non-default rule effort to use with the specified net. You can set any one of the options: • `auto`: follows the HARDSPACING for the NDR in the LEF file. • `hard`: makes the NDR hard even if HARDSPACING is not set in the LEF file. • `soft`: makes the NDR soft even if HARDSPACING is set in the LEF file. • `no_taper`: follows the HARDSPACING for the NDR in the LEF file and does not allow tapering at all pins. *Default*: `auto` *Data_type*: string, optional
`-shield_nets` *special_net_name*	Specifies a special net to use as a shield for a critical or high-speed net. Typically, you route shielded nets before routing other nets. You can specify one or two shield nets, however, you cannot control which side is used by the special net that is routed on the same layer as the signal routing. The shield terminates near the pin. Shielding information is saved in the DEF file. **Note:** The router does not support shielding nets that are defined in the SPECIALNET section of the DEF file. This limitation extends to nets that are defined partly in the SPECIALNET section and partly in the regular NET section of the file, such as some clock nets. If you set the shielding attribute on these nets, the router automatically sets the - skip_routing attribute on them as well. As a result, the nets are not modified by the router and, if they are not completely routed already, the router does not try to finish the connections. The router does not delete the shielding wires.
`-shield_side {one_side | two_sides}`	Specifies the number of sides to route shield. *Default*: `two_sides` *Data_type*: string, optional
`-si_post_route_fix {true | false}`	Repairs crosstalk violations on specified nets. *Data_type*: string, optional
`-skip_antenna_fix {true | false}`	Prevents the router from repairing violations on specified nets with process antenna violations. This attribute is useful on nets that will have antenna violations repaired at the next level of hierarchy. *Data_type*: string, optional
`-skip_routing {true | false}`	Prevents the NanoRoute router from routing or rerouting unrouted or partially routed nets. When you specify `-skip_routing` for a net, the router treats the net as if you had marked it + FIXED in the DEF file and it becomes an obstruction for other nets. If the net is partially routed, the router does not complete the routing. **Note**: The value you specify for this attribute also applies to Early Global Route. *Data_type*: string, optional
`-stack_distance` *distance*	Specifies the stacking distance in microns. *Default*: `0` *Data_type*: string, optional
`-stripe_layer_range` *min_layer*`:`*max_layer*	Specifies the layer range that accommodates different stripe_layer_range for different nets and subnets. You need to specify the following values with this parameter: • *max_layer*: Specifies the top-most floating stripe layer allowed for connection. • *min_layer*: Specifies the bottom-most floating stripe layer allowed for connection For example, 3:4 means that the constraint should be applied on M3 and M4. *Data_type*: string, optional

`-sub_net` *`sub_net_name`*	Specifies the subnet for which to set attributes. This parameter is used with the `-net` parameter.
	For example:
	`set_route_attributes -nets` *`netName`* `-sub_net` *`sub_net_name`*
	Note: Cadence recommends you to use the `set_db` command for changing the specified attribute value for a database object. The `set_db` command takes an object or object list as a starting point, then uses a period to traverse to other related objects, or access attributes. Additional periods can be used to continue traversing the database schema.
	For example, to specify the top preferred routing layer and bottom preferred routing layers attributes for a subnet, you can use the following `set_db` commands:
	`set_db [get_db [get_db head.libcells.name $cell_name -p].pgTerms.name $pg_pin_name -p].top_preferred_routing_layer [get_db head.layers.num 3 -p]`
	`set_db [get_db [get_db head.libcells.name $cell_name -p].pgTerms.name $pg_pin_name -p].bottom_preferred_routing_layer [get_db head.layers.num 2 -p]`
	instead of using the `set_route_attributes -sub_net` command.
	`set_route_attributes -nets VDD1 -sub_net VDD1_2 -inst_pin iA/FEAOB*:TVDD -top_preferred_routing_layer 3 -bottom_preferred_routing_layer 2`
	Data_type: string, optional

`-top_preferred_routing_layer` *`layer_number`*	
	Specifies the preferred highest layer for routing specified signal nets. This attribute is a soft limit; that is, NanoRoute might use a layer above the specified layer if necessary to complete routing. Use this attribute to improve timing by restricting the routing layers available for critical nets, while simultaneously routing other nets on all routing layers. *Range:* 0-15 **Note:** Layers can only be specified using their layer ID numbers. Use of LEF layer names is not supported by this parameter. *Data_type*: string, optional
`-top_preferred_shielding_layer` *`layer_number`*	
	Specifies the preferred highest shielding layer. **Note:** Layers can only be specified using their layer ID numbers. Use of LEF layer names is not supported by this parameter. *Data_type*: string, optional
`-weight` *`integer`*	Specifies a relative weight for the nets. This value is honored by placement and not by the router. *Default:* 2 *Data_type*: string, optional

`-reset`	Currently you cannot reset all the attributes of a net. You can reset each attribute of a net by using the `-reset` parameter with the attribute or you can use the “default” keyword for the value of each attribute.
	For example:
	• The following command sets the weight attribute of the net (scan_in_1) to 3 and the mask attribute to 2:
	`set_route_attributes -nets scan_in_1 -weight 3 -mask 2`
	• The following command resets the weight attribute of the net:
	`set_route_attributes -nets scan_in_1 -reset -weight -mask 2`
	• The following command resets the mask attribute of the net:
	`set_route_attributes -nets scan_in_1 -weight 3 -mask default`
	Data_type: string, optional

Examples

- The following command sets the net weight for `net1` to `4`:
  ```
  set_route_attributes -nets net1 -weight 4
  ```
  Setting the weight to `4` forces the router to route `net1` before it routes nets whose net weight is less than 4.
- The following command sets the net weight for all nets whose net name starts with `net1` to `4`:
  ```
  set_route_attributes -nets net1* -weight 4
  ```
- The following command limits routing for `net1` to layers Metal4 and Metal5 if possible:
  ```
  set_route_attributes -nets net1 -top_preferred_routing_layer Metal5 -bottom_preferred_routing_layer Metal4
  ```
- The following command adds two additional grid units around `net1`. The additional grid units do not have any routing.
  ```
  set_route_attributes -nets net1 -preferred_extra_space_tracks 2
  ```
- The following command uses the nondefault rule `wide_wire1` for routing `net1`:
  ```
  set_route_attributes -nets net1 -route_rule wide_wire1
  ```
- The following command uses the default rule for routing `net1`.
  ```
  set_route_attributes -nets net1 -route_rule default
  ```
- The following commands shield `net1` with both power and ground wires, and shield `net2` with a ground wire:
  ```
  set_route_attributes -nets net1 -shield_nets {vdd vss}
  set_route_attributes -nets net2 -shield_nets vss
  route_global_detail
  ```
- The following commands show how to shield two nets (do not shield more than one net with the same command):
  ```
  set_route_attributes -nets net1 -shield_nets abc_gnd
  set_route_attributes -nets net2 -shield_nets abc_gnd
  ```

Related Topics

- `get_route_attributes`
- `route_global_detail`
- Using the NanoRoute Router chapter in the *Innovus Stylus Common UI User Guide*

write_congested_area_report

```
write_congested_area_report
[-help]
out_file
[-all]
```

Writes the routing congestion information to an ASCII output file. The output file reports the coordinates of overflow cells, vertical tracks, and horizontal tracks. You can use the `write_congested_area_report` command after running Early Global Route.

Parameters

`-all`	Writes the congestion information for all gcells in the design to the specified output file. *Data_type*: bool, optional
out_file	Writes the congestion information for only the most congested gcells in the design to the specified output file. *Data_type*: string, required
`-help`	Outputs a brief description that includes the type and default information for each `write_congested_area_report` parameter. For a detailed description of the command and all of its parameters, use the man command `man write_congested_area_report`.

Example

- The following command writes the routing congestion information for all gcells to the `congest.txt` file:
  ```
  write_congested_area_report -all congest.txt
  ```

Related Information

- `report_congested_area`
- Using Early Global Route for Congestion and Timing Analysis chapter in the *Innovus Stylus Common UI User Guide*

write_litho_hotspots

```
write_litho_hotspots
[-help]
out_file filename
```

Writes out a HIF file after the NanoRoute router fixes lithography problems. The file reports the areas where the router made changes. Use this command with the `read_markers` command for incremental lithography checking and fixing.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `write_litho_hotspots` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man write_litho_hotspots`.
`out_file filename`	Specifies the HIF file to write out. *Data_type*: string, required

Example

The following example writes the HIF file `result.hif` marking the areas in which the NanoRoute router has made changes to fix lithography problems:

```
write_litho_hotspots out_file result.hif
```

Related Information

- `read_markers`
- Using Early Global Route for Congestion and Timing Analysis chapter in the *Innovus Stylus Common UI User Guide*
- Using the NanoRoute Router chapter in the *Innovus Stylus Common UI User Guide*

write_nets_in_congested_area

```
write_nets_in_congested_area
[-help]
out_file
```

Outputs the names of nets in the congested area of the design into the specified file, based on Early Global Route results. You can use the `write_nets_in_congested_area` command after running Early Global Route.

Parameters

out_file	Specifies the name of the file in which to output the net names. *Data_type*: string, required
-help	Outputs a brief description that includes the type and default information for each `write_nets_in_congested_area` parameter. For a detailed description of the command and all of its parameters, use the man command `man write_nets_in_congested_area`.

Related Information

- `route_early_global`
- Using Early Global Route for Congestion and Timing Analysis chapter in the *Innovus Stylus Common UI User Guide*

34

Signal Integrity Commands and Attributes

- analyze_ir_paths
- check_noise
- define_bundled_bus
- fix_ac_limit_violations
- get_glitch_threshold
- report_bundled_bus
- report_double_clocking
- report_instance_cdb
- report_noise
- report_noise_lib_pin
- report_noise_propagation
- report_voltage_scaling
- report_voltage_swing
- reset_glitch_derate
- reset_noise_lib_pin
- set_annotated_glitch
- set_cdb_binding
- set_glitch_derate
- set_glitch_threshold
- set_model_priority
- set_mutex_condition
- set_noise_lib_pin
- set_outbound_report
- set_quiet_aggressor
- si Category Attributes
- update_glitch

- write_spice_deck

analyze_ir_paths

```
analyze_ir_paths
[-help]
-clock_period <float>
[-hierarchical_activity_file <string>]
[-max_slack <float>]
[-pre_simulation_period <float>]
-seq_activity <float>
-simulation_period <float>
```

Generates a physical-aware multi-cycle switching activity file containing toggle information of voltage sensitive paths as well as timing critical paths in the design for Tempus Power Integrity analysis. This command generates ‘swsrc.txt’ file in the run directory, which then is fed back to the software using the `read_activity_file` command.

Parameters

`-help`	Prints out the command usage.
`-clock_period <float>`	Specifies the period of dominant clock of the design in library time units. Tempus PI refers to this specified period for multi-cycle activity generation.

`-hierarchical_activity_file <string>`	Provides a mechanism for different sequential activity for several partitions in the design. The following is the required format:
	`<hierarchical instance 1> <activity>`
	`<hierarchical instance 2> <activity>`
	…

`-max_slack <float>`	Specifies the maximum slack limit in library time units, above which the timing paths are not considered for activity generation.
`-pre_simulation_period <float>`	Specifies the pre-simulation duration that will be used by Tempus PI for activity generation. This parameter is useful when package model information is read-in using `set_package` command, as it allows the package oscillations to settle down. **Note**: The value specified must match to value specified under `set_rail_analysis_mode -record_results_start_time` setting.
`-seq_activity <float>`	Specifies the activity of sequential endpoints in the design. This value will be used to control the number of IR sensitive paths, along with their fanout cone, to toggle in each cycle.
`-simulation_period <float>`	Specifies the simulation period in library time units that Tempus PI dynamic power analysis will use. The larger the simulation period, the more will be the number of paths covered in the analysis. The following formula can be used to set the simulation period: *sim_cycles = (2/seq_act)*N* Where, • *seq_act* is the input sequential activity (e.g., 0.05, 0.1 etc.) • *N* is the number of nworst paths to be covered per endpoint • *sim_period* is equal to the sim_cycles*<time period of dominant clock> It is recommended to have N>=1, that is, at least one path per endpoint should be scheduled in the analysis.

Examples

- The following command generates the activity file for design with dominant clock of 2GHz, with input sequential activity being 0.1 and simulation duration being 20ns:

```
analyze_ir_paths -seq_activity 0.1 -simulation_period 20 -clock_period 0.5
```

- The following example specifies different sequential activity for different hierarchical partitions, with 0.2 sequential activity for unspecified partitions and top:

```
analyze_ir_paths -seq_activity 0.2 -simulation_period 30 -clock_period 0.5 -
hierarchical_activity_file hact.txt
Contents of 'hact.txt':
hierA/inst1 0.25
hierB/inst2 0.15
```

- The following command specifies the pre-simulation duration, particularly useful for package oscillations to settle when package models have been read-in:

```
analyze_ir_paths -seq_activity 0.1 -simulation_period 30 -clock_period 0.5 -
pre_simulation_period 10
```

check_noise

```
check_noise
[-help]
[-all]
[-check_only {receiver_no_propagation_model | receiver_no_dc_tolerence |
driver_no_current_source_model | driver_no_model}]
[-instance_list string]
[-instance_only {true | false}]
[-model_type {receiver | driver}]
[-pins string]
[-unique {true | false}]
[-verbose]
[-view string]
[ [ > | >> ] filename ]
```

Performs several consistency and completeness checks on noise models specified for a design. These checks help to ensure that the right set of models are available for the SI analysis engine to perform crosstalk noise analysis.

Parameters

`-help`	Prints out the command usage.
`-all`	Displays all the debug information.
`-check_only {receiver_no_propagation_model | receiver_no_dc_tolerence | driver_no_current_source_model | driver_no_model}`	Checks the warning name. The following warnings are checked: • `receiver_no_propagation_model`: Shows how many design pins are missing the receiver model for propagation. • `receiver_no_dc_tolerence`: Shows how many pins have no DC VH/VL tolerance specified. • `driver_no_current_source_model`: Shows how many design pins are missing the current source model. • `driver_no_model`: Specifies how many design pins are missing the driver model.
`-instance_list` *`string`*	Specifies the instance names.
`-instance_only {true | false}`	Prints each instance only once.
`-model_type {receiver | driver}`	Checks the presence of a valid model type - `receiver` or `driver` model. If the specified model is missing, a warning will be issued. The valid receiver models are the models having vivo tables, such as, CDB, CCSN, ECSMN. The valid driver models are CDB, CCS, ECSM, NLDM, and UDN.
`-pins` *`string`*	Specifies the instance pin names.
`-unique {true | false}`	Reports unique cells or pins for each power domain.
`-verbose`	Generates verbose details.
`-view` *`string`*	Specifies the view name.
`filename`	Re-directs output to a file.
`>>`*`filename`*	Appends to an output file.

Examples

- The following command generates the output as shown below:

```
check_noise -model_type receiver

   **WARN: (IMPESI-3125): AAE_WARN: The analysis view specified to check_noise is not found or
```

```
specified. Checking view default_emulate_view_setup.
-------------------------------------------------------------------------
CHECK NOISE SUMMARY
-------------------------------------------------------------------------
Warning Number of Warning
-------------------------------------------------------------------------
receiver_no_propagation_model 17931
receiver_no_dc_tolerence 17931

-------------------------------------------------------------------------
# CHECK NOISE WARNING DESCRIPTIONS:

# receiver_no_propagation_model: The receiver model for glitch
# propagation or ROP is not available, only RIP analysis will be performed.

# receiver_no_dc_tolerence: The input pin do not have vh/vl dc
# tolerence specified in the noise file. Default tolerences will be used.

-------------------------------------------------------------------------
```

Related Information

- check_timing
- set_glitch_threshold
- set_annotated_glitch

define_bundled_bus

```
define_bundled_bus
[-help]
[-add_members string]
[-delete_members string]
[-name string]
```

Allows you to control user-defined bus groups.

The software automatically identifies bus nets. However, in some cases the software may not be able to mark bus bits properly (or treat marked bits as bus). For improved bus identification, the `define_bundled_bus` command allows you to control the nets that you want to add (or remove) from the list of recognized bus bits.

Parameters

`-help`	Prints out the command usage.
`-add_members string`	Specifies the name of nets to be added to the group.
`-delete_members string`	Specifies the name of nets to be deleted from the group.
`-name string`	Specifies the name of the group.

Examples

- The following example command will create a bus named bus1 with 3 net members x[1] and x[2]:

```
define_bundled_bus -name bus1 -add_members {x[1] x[2]}
report_bundled_bus -type user_defined
############################################################

Summary:
Total number of user defined bus groups reported = 1
Total number of tool inferred bus groups reported = 0

------------------------------------------------------------

Group Name (user_defined) : bus1

Constituents:
- x[1]
- x[2]
```

- The following example command will add one more member x[3] and delete member x[2] to the bus 'bus1':

```
define_bundled_bus -name bus1 -add_members x[3] -delete_members x[2]

Added [1] net(s) to group 'bus1'.

Deleted [1] net(s) from user-defined group 'bus1'.
```

Related Commands

- report_bundled_bus

fix_ac_limit_violations

```
fix_ac_limit_violations
[-help]
[-exclude_io {true | false}]
[-exclude_nets_file filename]
[-fix_nets_category {data_only | clock_only | clock_and_data}]
[-fixing_method {route_rule down_size buffer}]
[-include_nets_file filename]
[-post_cts]
[-use_report_file filename]
[-use_segment_based_route_rule {true | false}]
```

Repairs violations associated with AC current density, wire self-heat, Joules heat, or electromigration, in a post-Route stage. Fixing methods are applying routing rules, downsizing drivers or adding a buffer to split up the net load on the violated nets. Use this command in conjunction with the `check_ac_limits` command. The `check_ac_limits` command marks the violating net for the `fix_ac_limit_violations` command to repair, and it can generate the report file that the `fix_ac_limit_violations` command can read.

The `fix_ac_limit_violations` command only applies a single fixing method to fix EM violations. Select the fixing method by specifying the `-fixing_method` parameter. If the method is not specified, the command will fix EM violations by setting route rules on violating routes. For example, the following usage will apply route rules for both clock and data nets while excluding nets with I/O pins:

```
fix_ac_limit_violations -fix_nets_category clock_and_data -exclude_io true
```

Before running `fix_ac_limit_violations`, you can generate an incoming EM violation report using the `check_ac_limits` command. This operation ensures that the appropriate data for `fix_ac_limit_violations` is included in the violation report. After that, this report should be provided as an input to `fix_ac_limit_violations` using the `-use_report_file` parameter. The following is an example of the sequence of operations:

```
check_ac_limits -report_db beforeFix.rpt -view hold_view

fix_ac_limit_violations -use_report_file beforeFix.rpt

check_ac_limits -report_db afterFix.rpt -view hold_view
```

By default, `check_ac_limits` calculates EM currents using the hold view. This means that `check_ac_limits` internally invokes a hold timing update. If it is desired to maintain the current setup timing graph in memory for use after EM fixing, the `-force_hold_view` parameter should be provided to `check_ac_limits` so that the hold timing update doesn't nullify the setup timing graph. Below is an example of this scenario:

```
time_design -post_route
check_ac_limits -report_db beforeFix.rpt -force_hold_view
fix_ac_limit_violations -use_report_file beforeFix.rpt
check_ac_limits -report_db afterFix.rpt -force_hold_view
```

Note: It is important to set the right switching activity factor on the nets before using the `check_ac_limits` command. Refer to the power Category

Attributes, `set_default_switching_activity` and `set_switching_activity` commands for further details on the default behaviors of the software and how to modify them. The example below shows how to set the global switching activity for verifying and fixing AC limit violations.

```
set_default_switching_activity -global_activity 0.2
propagate_activity -set_net_freq true
check_ac_limits -use_db_freq -report em.report
fix_ac_limit_violations -use_report_file em.report
```

Alternatively, `check_ac_limits -toggle` parameter can be used for more pessimistic analysis. In this case, modifying `set_default_switching_activity` is not needed. The following is an example of forcing every signal net to toggle at 50% of the clock rate:

```
check_ac_limits -toggle 0.5 -report_db em.report
fix_ac_limit_violations -use_report_file em.report
```

See `check_ac_limits` for further command parameters and usage**.**

Parameters

`-help`	Outputs a brief description that includes type and default information for each `fix_ac_limit_violations` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man fix_ac_limit_violations`
`-exclude_io {true | false}`	If set to `true`, excludes nets connected to I/O pins during violation fixing. *Default:* `false`
`-exclude_nets_file` *filename*	Specifies a file with the list of nets to exclude from AC limit violation repair.
`-fix_nets_category {data_only | clock_only | clock_and_data}`	Specifies fixing net types. For `data_only`, fixes data nets only. If `clock_only` is specified, fixes clock nets. If `clock_and_data` is specified, fixes clock and data nets. **Note**: For clock nets, only route rules will be used for fixing. Downsizing or buffering will not be used. *Default:* `data_only`

`-fixing_method {route_rule | down_size | buffer}`	Specifies the fixing method for correcting EM violations. You can specify any of the following: • `route_rule`: routing rules will be applied for fixing violations • `down_size`: driver cell downsizing will be applied for the violating nets • `buffer`: buffering will be applied to the violating nets *Default*: `route_rule`
`-include_nets_file` *filename*	Specifies a file that contains the list of nets for which to perform AC limit violation repair.
`-post_cts`	If specified, enables fix for the clock nets, for post-CTS DBs. This parameter should be used only with `-fix_net_category clock_only`. *Default*: `omitted`
`-use_report_file` *filename*	Reads the `check_ac_limits` report, and uses the violation information from the report file to set up proper routing rules. This is recommended for the best fixing performance.
`-use_segment_based_route_rule {true | false}`	Enables/disables segment-based route rules. If set to `true`, generates and applies segment-based route rules for fixing violations. This approach enables you to specify the desired width and distance for each layer of the violated net. *Default:* `true`

Examples

- The following command performs buffering only for the nets specified in the `netFile.txt` only::
  ```
  fix_ac_limit_violations -fixing_method buffer -include_nets_file "netFile.txt"
  ```
- To use downsizing, excluding nets with I/O pins:
  ```
  fix_ac_limit_violations -fixing_method down_size -exclude_IO true
  ```
- To fix both clock and data nets with route rules:
  ```
  fix_ac_limit_violations -fix_net_category clock_and_data
  ```
- The following example shows how to feed `check_ac_limits` report through the `-use_report_file` parameter.
  ```
  fix_ac_limit_violations -use_report_file em.report
  ```

Related Commands and Attributes

- `check_ac_limits`
- power Category Attributes
- `set_default_switching_activity`
- `set_switching_activity`
- `time_design`

Related Topics

- The following chapters in the *Innovus Stylus Common UI User Guide*:
  - Identifying and Viewing Violations
  - "Optimizing Signal EM Violations at PostRoute Stage" in the Optimizing Timing chapter

get_glitch_threshold

```
get_glitch_threshold
[parameter_names]
```

Displays information for the specified `set_glitch_threshold` parameters for the current session. If you do not specify a parameter, the software displays information for all `set_glitch_threshold` parameters.

Parameters

parameter_names	Displays information for the specified parameter. You can specify one or more parameters. See `set_glitch_threshold` for descriptions of the timing analysis mode parameters you can specify.

Examples

- The following command shows the failure threshold for ROP for single stage cells:
  ```
  get_glitch_threshold -failure_point last
  ```
  Here's the command output:
  ```
  value=0.4 -threshold_type reporting -glitch_type both -pin_type all -failure_point last -
  failure_dependency none
  ```

- The following command shows the failure threshold for the ROP for the internal net of multi-stage cells:
  ```
  get_glitch_threshold -failure_point internal
  ```
  Here's the command output:
  ```
  value=0.2 -threshold_type reporting -glitch_type both -pin_type all -failure_point internal -
  failure_dependency none
  ```
- The following command shows the failure threshold for the RIP for the input net of cells without CCSN/ECSMN/cdB
  ```
  get_glitch_threshold -failure_point input
  ```
  Here's the command output:
  ```
  value=0.5 -threshold_type reporting -glitch_type both -pin_type all -failure_point input -
  failure_dependency none
  ```

Related Commands

- set_glitch_threshold

report_bundled_bus

```
report_bundled_bus
[-help]
[-name string]
[-nets string]
[-type {user_defined | tool_inferred}]
[[ > | >> ]]
```

Generates a report for bundled buses.

Parameters

- help	Prints out the command usage.
-name *string*	Specifies the name of the user-defined bus group(s) to be reported.
-nets *string*	Specifies the name of the net(s) to be reported.
-type {user_defined | tool_inferred}	Reports the type of bundled bus(es) - user-defined or tool-inferred - to include in the report. By default, the report_bundled_bus command generates reports for both user-defined and software-inferred type of buses.
> *filename*	Redirects the output to the specified file.
>> *filename*	Appends to the specified output file.

Examples

- The following command shows the user-defined bus groups:

```
report_bundled_bus

Summary:
Total number of user defined bus groups reported = 3
Total number of tool inferred bus groups reported = 41

-----------------------------------------------------
Group Name (user_defined) : grp1

Constituents:
- EJ_PartNumber[15]
- EJ_PartNumber[14]
- EJ_PartNumber[13]
- EJ_PartNumber[12]
- EJ_PartNumber[11]
- EJ_PartNumber[10]
- EJ_PartNumber[9]
- EJ_PartNumber[8]
- EJ_PartNumber[7]
- EJ_PartNumber[6]
- EJ_PartNumber[5]
- EJ_PartNumber[4]
- EJ_PartNumber[3]
- EJ_PartNumber[2]
- EJ_PartNumber[1]
- EJ_PartNumber[0]
-----------------------------------------------------
Group Name (user_defined) : grp2

Constituents:
- EJ_ManufID[10]
- EJ_ManufID[9]
- EJ_ManufID[8]
```

Related Commands

- define_bundled_bus

report_double_clocking

```
report_double_clocking
[-help]
[-max_delta_delay float]
[-max_slew float]
[-nworst integer]
[-out_file string]
[-reverse_slope_limit float]
[-reverse_voltage_change_threshold float]
[-view string]
```

Reports noise impact on clock nets by generating information on the double clocking, transition type, and reverse slope.

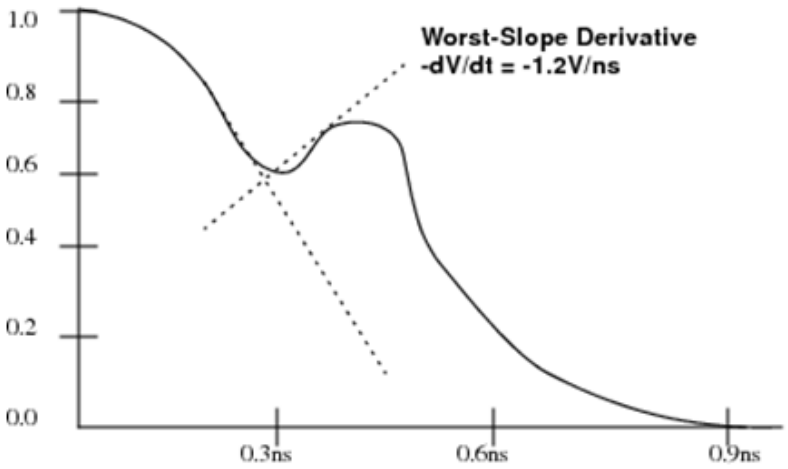
Note: The `report_double_clocking` command requires that clocks have been defined and applied prior to timing analysis with SI (`update_timing` or `report_timing`). The `report_double_clocking` command does not work with the glitch-only flow (`update_glitch`).

Double clocking occurs when the noise glitch attacks a transitioning signal. The multiple transition or double clocking check supplements the transition time check performed in static timing analysis. Transition time is defined as the duration of time that a signal takes to make the transition between two threshold points. However, transition time does not account for the waveform shape between the voltage thresholds.

For special signals such as clocks it is important to analyze the waveform shape between the voltage threshold in case the signal transitions slowly or in the opposite direction (even for a short duration of time) causing multiple clock edges. This command enables you to report the worst-case slope (dV/dtfor rising signal and -dV/dt for falling signal) of a signal transitioning between the voltage thresholds. The slope derivative is measured in units of volts per nanosecond (V/ns). This value can be positive if the signal does not change direction, 0 if the signal is flat, and negative if the signal changes direction even for a short duration of time.

The following figure shows two falling signal transitions:

Figure: Negative Worst-Slope Derivative



In the above figure, the falling signal transitions have identical transition times. However, the worst-case slope derivatives for the two falling waveforms are different. If a clock signal exhibits a negative slope derivative as in the case of Figure 13-2, there is a possibility of a double clocking functional failure. A higher reverse slope results in a higher possibility of a double clocking functional failure.

The double clocking report contains the following sections and their corresponding fields:

- **Double Clocking**
- Rise Slope (V/ns): Specifies the worst reverse slope of the rising clock signal as a measure of volt per nano-second.
- Fall Slope (V/ns): Specifies the worst reverse slope for the falling clock signal as a measure of volt per nano-second.
- VictimNet: Specifies the name of the victim net.
- **Transition Type**
- **Reverse Slope**

Parameters

`-help`	Prints out the command usage.
`-out_file <string>`	Specifies the output filename. *Default*: {$parm_filename_prefix}_clock.text
`-max_delta_delay float`	Reports clock nets, for which the delay change caused due to noise, exceeds the specified number. *Default*: 0
`-max_slew float`	Reports clock nets for which the transition time, including SI, exceeds the specified number. *Default*: 0
`-nworst integer`	Specifies the number of clock nets to report in each category. *Default*: 1
`-reverse_slope_limit float`	Reports the clock nets for which the slope is below the specified number. *Default*: 0

`-reverse_voltage_change_threshold` *float*	Controls the reverse slope failure criteria by allowing you to supply a minimum voltage change as a percentage of VDD. The reverse slope must be sustained in order for it to be considered as failing. *Default*: 0.1
`-view` *string*	Specifies the view name for which the report to be generated.

Examples

- The following command reports clock nets that have a reverse slope less than -1:
  ```
  report_double_clocking -reverse_slope_limit -1
  ```
- The following command reports 100 clock net violations of each category:
  ```
  report_double_clocking -nworst 100
  ```

Related Information

- `report_timing`
- `update_timing_condition`

report_instance_cdb

```
report_instance_cdb
[-help]
[-analysis_type {early | late}]
[objectListOrCollection]
[-out_file string]
[-quiet]
[-view string]
```

Allows you to report the cdB associated with all the instances or a subset of instances in a design.

Note: This command must be issued after `report_timing`, `update_timing`, or `update_glitch` commands.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `report_instance_cdb` parameter. For a detailed description of the command and all of its parameters, use the man command: `man report_instance_cdb`
`-analysis_type {early | late}`	Specify the analysis type for the query. *Default*: `late`
objectListOrCollection	Specify the instance for which the cdB association is desired. By default, the cdB association for all the instances is reported.
`-out_file` *string*	Specify the output filename for the instance report to go to. By default, the report output is displayed on the screen.
`-quiet`	Suppresses warning messages for cells missing in CDB files.
`-view` *string*	Specifies the view names. By default, values for all the views are reported.

Examples

- The following command reports the cdB instance connection for all the instances in the design and write it to the inst_cdb.rpt file:

```
> report_instance_cdb -out_file inst_cdb.rpt
```

- The following command reports the cdB instance connection for the u5 instance to the screen:

```
> report_instance_cdb u5

VIEW: default_emulate_view -- ANALYSIS TYPE: late
u5 : inv_t1 : test1.cdb.txt
```

- The following command reports the cdB instance connection for the early analysis type for the u5 and u6 instances

```
> report_instance_cdb -analysis_type early {u5 u6}

VIEW: default_emulate_view -- ANALYSIS TYPE: early
u5 : inv_t1 : test1.cdb.txt
u6 : inv_t2 : test2.cdb.txt
```

Related Commands

- report_timing

report_noise

```
report_noise
[-help]
[-cell_type {bbox | clock | data | latch}]
[-delay {max | min}]
[-double_clocking]
[-exclude_failures]
[-expand_virtual_attacker_bus]
[-failure]
[-format string]
[-gui_victim_nets_file]
[-histogram]
[-level {vl | vh | vlu | vho}]
[-max_threshold float]
[-merge_dir string]
[-nets string]
[-noisy_waveform]
[-output_format string]
[-out_file string]
[-overshoot_undershoot]
[-pins string]
[-precision integer]
[-quiet_nets]
[-report_virtual_attacker_constituents]
[-reverse_voltage_change_threshold <float>]
[-slack float]
[-skip_nets_or_pins_in_header]
[-sort_by string]
[-style {default | detailed}]
[-threshold float]
[-view string]
[[ > | >> ]]
```

Generates text-based noise reports.

Note: You need to ensure that the `si_glitch_enable_report` (to generate glitch report) or `si_delay_enable_report` (to generate detailed delay report) attributes are set to `true`, before the `report_timing`, `update_timing`, or `update_glitch` commands are run.

For more information on attributes, see si Category Attributes.

Parameters

`-help`	Prints out the command usage.

`-cell_type {bbox | clock | data | latch}`	Generates a glitch report for the user-specified cell-type. You can specify one of the following options: • `bbox`: Generates a report for only the black-box receivers • `clock`: Generates a report for only the clock pin receivers • `data`: Generates a report for only the data pin receivers • `latch`: Generates a report for only the latch pin receivers
`-delay {min | max}`	Reports delay uncertainty instead of noise values. The only legal values are `min` or `max` for minimum or maximum path delay analysis. The `report_noise -clock -delay min | max` command outputs minimum/maximum delay report for clock nets.
`-double_clocking`	Enables double clocking reporting.
`-exclude_failures`	Excludes all failing nets from the glitch report.
`-expand_virtual_attacker_bus`	Reports virtual attacker components and expands bus bits in the noise reports. This parameter works only when the bus virtual attacker feature is enabled using the `set_db si_enable_virtual_attacker_constituent_report true` setting.

`-failure`	Reports receiver peak failures and DC tolerance failure nets. Use this option only for glitch noise failure reports.
	Option: Directory Name:
	`-sort_by noise glitch`
	`-sort_by rcvr_peak glitch`

`-format` *string*	Specifies the output glitch report format fields. The valid fields are: arc, attacker_edge, attacker_logical_correlation_phase, attacker_net, attacker_offset, attacker_peak, attacker_slew, attacker_source, attacker_status, attacker_vdd, attacker_xcap, cell_type, delay, glitch_slack, incr_delay, level, peak, receiver_cell, receiver_input_failure_threshold, receiver_input_pin, receiver_output_failure_threshold, receiver_output_peak, slack, slack_status, stage_info, thresh_source, verbose_logical_correlation_report, victim_edge, victim_library, victim_logical_correlation_phase, victim_net, victim_slew, victim_total_area, victim_total_cap, victim_total_cross_cap, victim_vdd, victim_width

Each field corresponds to the following columns of the glitch report:

- arc: Arc
- attacker_edge: Edge
- attacker_net: Net
- attacker_offset: Offset (ps)
- attacker_peak: Peak (mV)
- attacker_slew: Slew (ps)
- attacker_source: Constituents: Source
- attacker_status: Statue
- attacker_vdd Vdd (V)
- attacker_xcap: Xcap (fF)
- glitch_slack: Slack
- level: Level
- peak: Peak (mV)
- receiver_cell: Receiver cell
- receiver_input_failure_threshold: Receiver Input Threshold
- receiver_input_pin: ReceiverNet
- receiver_output_failure_threshold: Failure threshold
- receiver_output_peak: Receiver output peak Value
- stage_info
- thresh_source: ReceiverThresholdType (Global | User | Library)
- victim_library: Library
- victim_net: VictimNet
- victim_total_area: TotalArea
- victim_total_cap: TotalCap (fF)
- victim_vdd: Vdd (V)
- victim_width: Width (ps)

The following options are not available with the single line format:

- delay

	• incr_delay • slack • thresh_source • victim_edge • victim_slew • victim_total_cross_cap To choose whether the report format should be on a single line or multiple lines, you can use the `-output_format` parameter.
`-gui_victim_nets_file`	Generates the GUI format glitch reports (`aae or aae_<viewname>` files under the reporting directory) and incremental delay reports (`aae_<max/min>_incDelay or aae_<viewname>_<max/min>_incDelay` files under the reporting directory) to be used in the Display Noise Net form.
`-histogram`	Generates the noise histogram.
`-level {vl | vh | vlu | vho}`	Specifies the level for which a report needs to be generated when single-level report is desired. You can specify one of these options - vl, vh, vlu, and vho.
`-max_threshold` *float*	Specifies the maximum peak noise or receiver peak thresholds to report on.
`-merge_dir` *string*	Accepts a directory name and generates a tool compatible database.
`-nets` *string*	Reports the specified nets in the noise report.
`-noisy_waveform`	Generates a report for bumpy nets.
`-out_file` *string*	Specifies the name of the output file name prefix. The name of the output file is generated by combining the prefix and the specified view name. If you do not specify this option: • The software automatically names the text report as 'glitch' for noise glitch reports and 'delay' for noise delay reports. • The software dumps the generated report to the console window for better usability. **Note**: If the text report file with the `.gz` extension is used, the output file is gzipped to generate a compressed report.
`-pins` *string*	Specifies the name of pins to be reported.

`-precision` *integer*	Specifies the precision value up to which the glitch data is to be printed. The allowed values are 0, 1, 2 and 3.
`-output_format` *string*	Specifies if the output format should be on a single line or on multiple lines. You can specify one of the following options: • `single_line` • `multiple_line` If the `single_line` option is specified and the attacker info is chosen as a field, then all the attacker related format options will be printed together. The `-output_format` parameter only works with the `-format` option and does not affect the default reporting output format. *Default*: `multiple_line`
`-overshoot_undershoot`	Generates an overshoot/undershoot glitch report. *Default*: If this option is not used during glitch reporting, then VL and VH type of glitches will be reported.
`-quiet_nets`	Generates the quiet attackers report.
`-report_virtual_attacker_constituents`	Generates reports for virtual attacker constituents.
`-reverse_voltage_change_threshold <float>`	Reports the clock nets with reverse swing greater than the user-specified threshold value (ratio of VDD). By default, the threshold value is set to 0.1.
`-slack` *float*	Generates a report for bumpy nets based on slack.
`-skip_nets_or_pins_in_header`	Generates the glitch report without a list of nets or pins in the report header.

`-sort_by` *string*	Requests the sorting of entries. You can sort by: • `noise` • `receiver_peak` • `slack` • `cell_type` Innovus determines the default HTML output directories based on the options you choose. Depending on the sorting option you select, Innovus sorts the report fields in the following order: **-sort_by Option Level 1 Level 2 Level 3 Level 4** rcvr_peak receiver peak noise netname level noise noise - netname level
	For example, if you use the `-sort_by noise` option, the first level of sorting will be based on noise, followed by receiver peak, netname, and level. **Note**: The report displays the `vl` noise level before the `vh` noise level. *Default*: `receiver_peak`
`-style {default | detailed}`	Generates reports in one of the styles - `extended` or `default`. • `detailed`: Generates detailed reports with victim and aggressor timing windows, dropped aggressor slews, and reasons for dropped aggressors. Additionally, the report also shows driver pins and driver library pins. • `default`: Generates the default report, which is similar to the text report generated using the `-text` parameter.

`-threshold` *float*	Specifies the minimum peak noise, sensitivity, absolute delay uncertainty, relative delay uncertainty, and receiver peak thresholds to report on. *Default*: • 0.10 of vdd for nets - when the glitch report is generated using the `report_noise -sort_by noise` option. • 10 (ps) for nets - when a detailed delay report is generated using the `report_noise -delay` parameter). **Note**: • The `-threshold` parameter accepts percentage values. • All `-sort_by` values that represent voltages will be in mV.
`-view` *string*	Specifies the view name.
filename `[[ > | >> ]]`	Redirects output to the specified file.

File and Directory Naming Convention for Reports

The software uses a set of naming conventions for the report files and directories that are generated by the `report_noise` command. Each type of report file or directory is identified by a combination of strings that are attached to its name. The software adds the appropriate string to the report file or directory name.

Noise Reports

For noise reports, the `report_noise` command uses the following naming convention for report files or directories:

```
glitch
```

Delay Reports

For delay reports, the `report_noise` command uses the following naming convention for report files or directories:

```
delay
```

Report File or Directory Name Examples

This section contains examples of using the `report_noise` command with different options to generate different types of reports and depicts the corresponding impact on the name of the report file or directory.

Examples of Generating Noise Report Files or Directories

The following table shows the combination of different options used with the `report_noise` command for noise reports and the corresponding names of the generated report files or directories:

Command(s)	Report File/Directory Name
`report_noise -sort_by noise -out_file example_glitch.txt`	`example_glitch.txt`
`report_noise -sort_by rcvr_peak -out_file example_recevier_peak.txt`	`example_receiver_peak.txt`
`report_noise -failure -out_file failure.txt`	`failure.txt`

Examples of Generating Delay Report Files or Directories

The following table shows the combination of different options used with the `report_noise` command for delay reports and the corresponding names of the generated report files or directories:

Command(s)	Report File/Directory Names
`report_noise -delay min -out_file min_delay.txt`	`min_delay.txt`
`report_noise -delay max -out_file max_delay.txt`	`max_delay.txt`

Sample Delay Report

```
Type  IncrementalDelay(ps)  Delay(NomMax/Max)  VictimNet
R     1725.800              128.300/1854.100   inst1/I {victim}

Constituents:
       Offset(ps)           Slew(ps)   Xcap(fF)    Edge   Net
Cpl:   3707.803             3592.375   48.692      F      att1
Cpl:   1028.237             967.125    6.592       F      att2
Cpl:   1055.632             963.625    4.348       F      att3
Cpl:   885.057              744.750    4.330       F      att4
```

Examples

- The following command reports noise for the view named default_emulate_view:

  ```
  report_noise -delay max -threshold 1600 -out_file delay_max_format.rpt -format {attacker_slew
   attacker_xcap  attacker_status attacker_edge   attacker_net stage_info slack  victim_slew
   victim_total_cross_cap victim_net }
  ```

- The following command reports noise for the view named `default_emulate_view`:

```
report_noise -delay max -threshold 1600 -out_file delay_max_format.rpt -format {attacker_slew
 attacker_xcap  attacker_status attacker_edge   attacker_net stage_info slack  victim_slew
 victim_total_cross_cap victim_net }

****************************************************
Noise Delay Report (for view default_emulate_view)
****************************************************

Report Options:
-----------------------------------------------------
-type delay
-threshold 1600 (ps)
-delay max
[-style default]
[-sortBy IncrDelay]
-----------------------------------------------------
-----------------------------------------------------
Slack(ps)   Slew(ps)   Total    Xcap(fF)    VictimNet
-2.973      0.635      277.574  seg3/u6/A   {seg3/n5}

Stage information:
            Cell       Pin
Driver:     INV        seg3/u5/Y
Receiver:   INV        seg3/u6/A

Constituents:
Slew(ps)   Xcap(fF)   Status   Edge   Net
331.500    138.787    INF      F      seg1/n5
358.667    138.759    SY       F      seg2/n5
-----------------------------------------------------
Slack(ps)  Slew(ps)   Total Xcap(fF)  VictimNet
14.906     0.597      278.115         seg1/u6/A {seg1/n5}

Stage information:
           Cell       Pin
Driver :   INV        seg1/u5/Y
Receiver : INV        seg1/u6/A

Constituents:
Slew(ps)   Xcap(fF)   Status   Edge   Net
279.500    139.280    INF      F      w3
359.333    138.807    INF      F      seg3/n5
-----------------------------------------------------
Slack(ps) Slew(ps)    Total Xcap(fF)  VictimNet
15.737    0.587       270.604         seg1/u12/A {seg1/n11}

Stage information:
           Cell    Pin
Driver :   INV     seg1/u11/Y
Receiver : INV     seg1/u12/A
```

```
Constituents:
Slew(ps)   Xcap(fF)   Status   Edge   Net
276.500    135.654    INF      F      w5
337.167    134.517    INF      F      seg3/n11
```

- The following command reports noise for the specified reporting fields:

```
report_noise -delay max -format {delay incr_delay victim_slew victim_total_cap
victim_total_cross_cap victim_vdd victim_net attacker_offset attacker_slew attacker_xcap
attacker_vdd attacker_edge attacker_status attacker_net} -out_file custom.rpt
------------------------------
Delay(NomMax/Max) IncrDelay(ps) Slew(ps) TotalCap(fF) Total Xcap(fF) Vdd(V) VictimNet

-43.400/2418.200  2461.600      12.372   639.088      599.796        1.62   i1/A {n1}

Type  IncrDelay(ps) Delay(NomMax/Max)  VictimInst

R     3253.300      5457.100/8710.400  drv/Y {INV:A}

Constituents:
Offset(ps)  Slew(ps)  Xcap(fF)  Vdd(V)  Edge  Status  Net
-245.000    30.000    52.145    1.62    F     att1
-199.000    120.000   26.751    0.00    F     INF     ~n1
------------------------------
```

- To generate a glitch report, you can use the following command:

```
report_noise -out_file glitch.txt
```

- To generate a glitch report sorted by input noise, you can use the following command:

```
report_noise -out_file sort_by_glitch_input.txt -sort_by noise
```

- To generate a glitch failure report, you can use the following command:

```
report_noise -out_file glitch_failure.txt -failure
```

- To generate a glitch report for VL noises sorted by input noise, you can use the following command:

```
report_noise -sort_by noise -level vl -out_file vl_glitch.txt
```

- To generate a glitch report for only latch receiver pin cell types, you can use the following command:

```
report_noise -out_file latch_glitch.txt -cell_type latch
```

Related Information

- report_timing
- update_timing
- report_delay_calculation

report_noise_lib_pin

```
report_noise_lib_pin
[-help]
[> filename]
```

Generates a report providing information on all the user-specified noise mappings in the design.

Parameters

`-help`	Prints out the command usage.
`> filename`	Writes the report to the specified file name. If the file already exists, the software will overwrite it.

Examples

- The following command generates the output report 'noise_mapping.rpt' which contains all the noise mappings present in the design:

  ```
  report_noise_lib_pin > noise_mapping.rpt
  ```

  noise_mapping.rpt

  ```
  ---------------------------------------------------
  from_pin                    to_pin
  ---------------------------------------------------
  library1/inverter/I         library2/macro/A
  library1/buffer/I           library3/memory/ADD[0]
  ---------------------------------------------------
  ```

Related Commands

- set_noise_lib_pin
- reset_noise_lib_pin

report_noise_propagation

```
report_noise_propagation
[-help]
[-level {VL | VH | worst | all}]
[-max_paths integer]
[-max_slack float]
[-nets string]
[-nworst integer]
[-pins string]
[-source_depth integer]
[-views string]
[ [ > | >> ] ]
```

Prints text-based noise propagation reports that show glitch-source tracing for all the user-specified or all-violating receiver pins. The report allows you to view full paths showing the glitch propagating from the source to the violating receiver pin. This helps to easily identify and fix glitch violations.

Note: You need to set `si_glitch_enable_report` attribute to `true` to enable glitch reporting.

Parameters

`-help`	Prints out the command usage. For a detailed description of the command and all of its parameters, use the `man` command: `man report_noise_propagation`
`-level {VL | VH | worst | all}`	Reports paths for specified glitch level. You can specify one of these options - VL, VH, worst, or all.
`-max_paths integer`	Reports the specified number of paths.
`-max_slack float`	Reports paths with glitch slack equal to or lesser than the specified value.
`-nets string`	Reports paths with endpoint being all the input pins connected to the specified net.
`-nworst integer`	Specifies the maximum number of sources per endpoint for which paths need to be reported.
`-pins string`	Reports the paths for the specified endpoint pins.
`-source_depth integer`	Specifies the maximum number of sources to be traced in any given branch in the fanin cone of violation.
`-views string`	Reports the paths for the specified view.
`> filename`	Redirects output to the specified file.
`>> filename`	Appends output to the specified output file.

Related Information

- report_noise

report_voltage_scaling

```
report_voltage_scaling
[-help]
objectListOrCollection
[-analysis_type {early | late}]
[-view string]
[-out_file string]
```

Reports instances whose timing data is interpolated due to voltage mismatch. When a cell is operating in the design at a voltage different from voltages at which it is characterized in the timing libraries, delay calculation (DC) interpolates/derives the timing data at the operating voltage.

Parameters

`-help`	Prints out the command usage.
`-analysis_type {early | late}`	Specifies whether early or late views have to be checked for voltage scaling. If unspecified, only 'late' views are checked.
`-out_file` *string*	Specifies the name of the output file to which the report will be written. When the specified file name contains a directory path, the directory hierarchy is automatically created, if not present. If this option is not specified, the report is written to standard output (screen).
objectListOrCollection	Specifies the name(s) or a collection of leaf-level instance(s) for which voltage scaling needs to be checked and reported. If not specified, all instances of the design are checked.
`-view` *string*	Specify the name(s) of view(s) to be checked for voltage scaling. If unspecified, all active analysis views are checked.

Examples

The following example shows a sample `report_voltage_scaling` report:

```
report_voltage_scaling i_6 -analysis_type late -view func_worst_cmax

VIEW: func_worst_cmax -- ANALYSIS TYPE: late
<pin-name> : <cell-name> : <actual-voltage> : <low-lib-name> : <high-lib-name>
i_6/Y : BUFX5MTH : 0.820 : elc_08v : elc_09v
```

The report is generated with the `func_worst_cmax` view. You can use the `-view` parameter to report for other views. You can run the `report_analysis_views` command to list views.

Related Information

- report_analysis_views

report_voltage_swing

```
report_voltage_swing
[-help]
-high_threshold float
-low_threshold float
[-out_file string]
[-required_width {all | clock}]
[-view string]
```

Reports the voltage swings that are shorter than the user-specified thresholds for both clock and data signals.

Parameters

`-help`	Outputs a brief description of the `report_voltage_swing` command. For a detailed description of the command, use the man command: `man report_voltage_swing`
`-high_threshold float`	Specifies the high threshold percentage of the pin supply voltage.
`-low_threshold float`	Specifies the low threshold percentage of the pin supply voltage.
`-out_file string`	Specifies the name of the output file.
`-required_width {all | clock}`	Reports the required high and low pulse widths corresponding to the user-specified high and low thresholds. You can specify one of the following options: • `clock`: Reports the required widths for all clock pins in the design. • `all`: Reports the required pulse widths for all input pins in the design. *Default*: `all`
`-view string`	Specifies the name of the view being defined. **Note**: If this option is not specified, it will dump report for all views.

Examples

An output file in the text format containing violating pin information is presented below:

```
View Name: default_setup_view1
Hpin     Type_of_Voilation   Vdd    Vss    Slews(Rise/fall)  Margin (threshold)
X/buf/A (upper/lower)       0.9v   0.0V   210ps/240ps       745mv (810mv)
```

reset_glitch_derate

```
reset_glitch_derate
[-help]
[-derate_height]
[-derate_width]
[-glitch_type string]
[-instance_pin string]
[-offset]
[-pin string]
[-view string]
```

Removes the derating factors for glitch waveforms that were previously set using the `set_glitch_derate` command.

Parameters

`-help`	Prints out the command usage.
`-derate_height`	Resets the derate height.
`-derate_width`	Resets the derate width.
`-glitch_type string`	Specifies the glitch type vl, vh, or both.
`-instance_pin string`	Specifies the instance pin names on which derate factors are to be reset.
`-offset`	Resets the derate offset.
`-pin string`	Specifies the library pin names on which derate factors are to be reset.
`-view string`	Specifies the view name.

Related Commands

- `set_glitch_derate`

reset_noise_lib_pin

```
reset_noise_lib_pin
[-help]
[-all]
[-to string]
```

Allows you to reset any noise property mapping, specified previously using the `set_noise_lib_pin` command.

Parameters

`-help`	Prints out the command usage.
`-all`	Resets all the noise mappings in the design.
`-to string`	Specifies the library pin name for which the user-specified noise mapping (if any) needs to be reset. **Note**: The `-to` parameter accepts library cell pin names (that can be obtained using the `get_lib_pins` command). This parameters does not honour collections or instance pin names.

Examples

- The following command resets any noise mapping previously applied to the library pin library1/macro_cell/A :
  ```
  reset_noise_lib_pin -to library1/macro_cell/A
  ```
- The following command resets all the existing noise mappings in the design:
  ```
  reset_noise_lib_pin -all
  ```

Related Commands

- `set_noise_lib_pin`
- `report_noise_lib_pin`

set_annotated_glitch

```
set_annotated_glitch
[-help]
[-ports string]
[-voltage_high string]
[-overshoot string]
[-view string]
[-voltage_low string]
[-undershoot string]
```

Allows the ability to annotate glitches at any specified pin/port. This command setting will override the VH/VL glitch with the specified value on the given point and use that glitch value to propagate downstream.

If the defined annotated glitch waveforms do not start and end at 0, then the software will not consider those waveforms and a warning will be issued. In the following example both VL and VH waves should start and end with 0.0 values. Since the VH wave is not ending at 0, this waveform will be ignored.

```
set_annotated_glitch -vl_glitch {0 0 100e-12 0.01 200e-12 0.04 300e-12 0.03 600e-12 0} -port
dff1/CK -vh_glitch {0 0 100e-12 0.1 200e-12 0.4 300e-12 0.3 600e-12 0.1} .
```

The software does not save (and report) waveforms below 5% of VDD. Consider for example, the annotated glitch defined on a pin as:

```
set_annotated_glitch -port dff1/CK -vh_glitch {0 0 100e-12 0.01 200e-12 0.04 300e-12 0.03 600e-12
0}
```

In this case, the glitch peak is 0.04 (40mV), which is below 5% of VDD (1080mV) - equal to 54mV. Since this annotated glitch waveform peak is very low, the `report_noise` command will not report it. However, if the failure criteria is lower than 5%, then the software will save and report small glitch violations.

Parameters

`-help`	Prints out the command usage.
`-ports` *string*	Specifies port or pin names on which you need to apply glitches. This parameter works only when `si_enable_glitch_propagation` attribute is set to `true` in the `report_timing` flow. In the `update_glitch` flow, the `si_enable_glitch_propagation` attribute is already set to `true`.
`-voltage_high` *string*	Specifies a list of VH glitch wave points. You must specify at least 3 time value pairs.
`-overshoot` *string*	Specifies a list of VHO glitch wave points.
`-view` *string*	Specifies a list of view names.
`-voltage_low` *string*	Specifies a list of VL glitch wave points. You must specify at least 3 time value pairs.
`-undershoot` *string*	Specifies a list of VLU glitch wave points.

Examples

- The following example annotates the specified VL glitch wave points:
```
set_annotated_glitch -vl_glitch {0 0.0  0.2e-9 0.540 0.405e-9 0} -vh_glitch {0.1e-9  0
0.205e-9 0.50 0.405e-9 0.200  0.6e-9 0.0} -port {A}
```
- The following command overrides the glitch value at RIP with the annotated glitch, because glitch is annotated at the receiver input:
```
set_annotated_glitch -vl_glitch {0 0 100e-12 0.1 200e-12 0.4 300e-12 0.3 600e-12 0} -port
seg2/u9/D
update_glitch
report_noise -nets seg2/n8

-------------------------------------------------------------------------
Peak(mV) Level TotalCap(fF) TotalArea Width(ps) Vdd(mV) Library VictimNet
399.924  VL    144.11       110.00    550.104   1080    cell_w  seg2/u9/D {seg2/n8}

Receiver Input Threshold:
Value    (Threshold)  ReceiverNet Receiver failure_type
399.924  (431.244)    seg2/u9/D   (DFF)    [bbox]

Constituents:
Source Peak(mV) Offset(ps) Slew(ps) Xcap(fF) Vdd(mV) Edge Status Net
```

```
Prp:            399.924    --       --       --      --   --     seg2/u8/Y
-------------------------------------------------------------------------------
```

- In the following example, the impact of aggressors coupled with the victim stage (along with the annotated glitch) is also considered because annotation is done at the driver output:

```
set_annotated_glitch -vl_glitch {0 0 100e-12 0.1 200e-12 0.4 300e-12 0.3 600e-12 0} -port
seg2/u8/Y
update_glitch
report_noise -nets seg2/n8

-------------------------------------------------------------------------------
Peak(mV) Level TotalCap(fF) TotalArea Width(ps) Vdd(mV) Library  VictimNet
548.208  VL    144.11       159.30    581.172   1080    cell_w   seg2/u9/D {seg2/n8}

Receiver Input Threshold:
Value (Threshold) ReceiverNet Receiver failure_type
548.208 (432.648) seg2/u9/D   (DFF)    [bbox]

Constituents:
Source Peak(mV) Offset(ps) Slew(ps) Xcap(fF) Vdd(mV) Edge  Status  Net
Cpl:           80.148     371.000  340.000  68.138  1080  R ACT   seg3/n8
Cpl:           75.159     381.000  348.500  62.007  1080  R ACT   seg1/n8
Prp:           392.904    --       --       --      --    --      seg2/u8/Y
-------------------------------------------------------------------------------
```

Related Commands

- report_noise
- update_glitch

set_cdb_binding

```
set_cdb_binding
[-help]
-from string
-to string
```

Provides an overriding mechanism that allows overriding a UDN to Liberty mapping at the command file level.

This command helps to correct any mismapped cases by enabling the software to select a user-specified cdB to Liberty mapping.

Parameters

`-help`	Prints out the command usage.
`-from string`	Specifies name of the the CDB file and the cell name.
`-to string`	Specifies the name of the library file.

Examples

- When loading multiple cdB's with the same PVT, but different secondary voltages, you can use the `set_cdb_binding` command to ensure that they get bound correctly, as shown below:

```
set_cdb_binding -from pvt1_1.cdb -to pvt1_1.lib
set_cdb_binding -from pvt1_2.cdb -to pvt1_2.lib
```

set_glitch_derate

```
set_glitch_derate
[-help]
[-derate_height float]
[-derate_width float]
[-glitch_type string]
[-instance_pin string]
[-offset float]
[-pin string]
[-view string]
```

Sets the derating factors for glitch waveforms and determines the effect of on-chip variation on noise.

Parameters

`-help`	Prints out the command usage.
`-derate_height float`	Specifies the derate height value. This parameter scales all the glitch waveforms on the specified pins. A value of 1.0 indicates that no glitch waveform derating will take place. A value greater than 1.0, such as 1.1, indicates a 10 percent increase in the magnitude of the complete glitch waveform, which results in an increase in propagated noise and noise failure. A value less than 1.0 reduces the magnitude of the complete glitch waveform.
`-derate_width float`	Specifies the derate width value.
`-glitch_type string`	Specifies the glitch type - vl, vh, or all.
`-instance_pin string`	Specifies the instance pin names on which derate is to be applied.
`-offset float`	Specifies the derate offset value.
`-pin string`	Specifies the library pin names on which derate value is to be applied.
`-view string`	Specifies the name of the view.

Related Information

- reset_glitch_derate
- update_timing_condition

set_glitch_threshold

```
set_glitch_threshold
[-help]
[-cells string]
[-failure_dependency {none | input | last}]
[-failure_point {input | internal | last}]
[-glitch_type {both | vl | vh | vlu | vho | vl_vh | vlu_vho}]
[-library_pins <string>]
[-nets string]
[-pins string]
[-pin_type {clock | data | latch | all}]
[-threshold_type {failure | reporting | propagation}]
-value float
[-views string]
```

Allows for additional control of failure criteria for SI glitch analysis.

Parameters

`-help`	Prints out the command usage. For a detailed description of the command and all of its parameters, use the `man` command: `man set_glitch_threshold`.
`-cells` *`string`*	Specifies list of library cell names.
`-failure_dependency {none | input | last}`	Specifies the failure dependency. *Default*: `none`
`-failure_point {input | internal | last}`	Specifies where the failure or reporting criteria applies. ◦ `none`: Criteria does not apply. ◦ `input`: Threshold applies when the failure point is at the input pin of the receiver cell. ◦ `internal`: Threshold is applied when the failure point is internal to the cell. This occurs for multi-stage cells. The failure point will be the output of the first stage inside the cell. ◦ `last`: Threshold is applied when the failure point is the output of the cell. This occurs for single-stage cells, such as, inverters. *Default*: `internal` and `last`

`-glitch_type {both | vl | vh | vlu | vho | vl_vh | vlu_vho}`	Specifies the glitch type the command applies to. • `both`: Refers to `vl` and `vh` type of glitches. • `vl_vh`: Includes VL and VH noise. • `vlu_vho`: Includes VLU and VHO noise. *Default*: `both`
`-library_pins` *string*	Specifies a list of library pin names.
`-nets` *string*	Specifies the list of nets the command applies to. Default is not net-specific.
`-pins` *string*	Specifies the list of pins the command applies to. Default is not pin-specific.
`-pin_type {clock |data | latch |all}`	Specifies the pin type the command applies to. *Default*: all
`-threshold_type {failure | reporting | propagation}`	Specifies whether the command options apply to failure, reporting, or propagation criteria. In case of VL/VH glitch analysis, it is mandatory to specify the `-threshold_type` parameter. This is because when the computed glitch (ROP) at any given stage exceeds the failure threshold (when glitch propagation is enabled), then it is not propagated and is flagged as failure in the noise report. With the `-threshold_type failure` setting, the propagation will stop as per the new threshold, and the failure flagging will change accordingly. However, with the `-threshold_type reporting` setting, the propagation threshold remains unchanged, but the failure reporting is updated. When `-threshold_type propagation` is used, the software allows for failing nets with ROP values less than the propagation threshold to propagate through single stage cells. In case of VLU/VHO glitch analysis, it is optional to use the `-threshold_type` parameter because the above flow is not applicable. *Default*: `failure`
`-value` *float*	Specifies the value for the threshold as a ratio of the local VDD. *Default*: 0.1
`-views` *string*	Specifies a list of view names.

Examples

Here are some examples.

- The following command moves the failure point from ROP to RIP for `inst1/a1` and uses a RIP failure of *0.2*ReceiverVDD*:

  ```
  set_glitch_threshold -failure_point input -pin inst1/a1 -threshold_type failure -value 0.2
  ```

- The following command sets clock failure criteria to 0.1*VDD, latch failure criteria to 0.2*VDD and all other nets to 0.3*VDD (set before update_glitch):

  ```
  > set_glitch_threshold -threshold_type failure -pin_type all -failure_point last -value 0.3
  > set_glitch_threshold -threshold_type failure -pin_type clock -failure_point last -value 0.2
  > set_glitch_threshold -threshold_type failure -pin_type latch -failure_point last -value 0.1
  ```

- The following command sets the failure threshold at the output of the 1st channel connect inside all the instances of macro1 to be 0.2 (set before update_glitch):

  ```
  > set_glitch_threshold -threshold_type failure -cell {macro1} -failure_point internal -value
  0.2
  ```

- The following command changes the reporting criteria for latches to be 0.15*VDD (set after `update_glitch`):

  ```
  > set_glitch_threshold -threshold_type reporting -pin_type latch -value 0.15
  ```

  If you need to set all the data failure criteria to be 0.3, then you need to make the following settings:

  ```
  > set_glitch_threshold -failure_point internal \
  -threshold_type failure -value 0.3 \
  -pin_type data

  > set_glitch_threshold -failure_point last \
  -threshold_type failure -value 0.3 \
  -pin_type data
  ```

Related Information

- `update_timing_condition`

set_model_priority

```
set_model_priority
[-help]
[-objects string]
[-priority_list string]
```

Allows you to define the priority order of models.

Parameters

`-help`	Prints out the command usage
`-objects string`	Specifies a list of library cells or library pins for which priority is specified.
`-priority_list string`	Specifies a list of pin model priority. Default: `{ecsm ccs nldm cdb udn}`

Examples

- To force CCSN to be used, you can set `timing_library_read_ccs_noise_data` attribute to `true` before reading the timing libraries and then use the following command before using the `update_glitch` command:

  ```
  set_model_priority -priority_list ccs
  ```

Related Information

- `update_timing_condition`

set_mutex_condition

```
set_mutex_condition
[-help]
[netNames]
```

Allows the software to use one of the specified attackers - each attacker is evaluated and the worst one (for each victim net) is selected.

This command applies a mutex group to a set of attackers (for a victim) to reduce pessimism. A mutex group is a group of nets of which only one can transition in a given cycle.

Parameters

`-help`	Prints out the command usage.
netNames	Specifies the net names of the mutex group.

Examples

- The following example shows that only one of the specified nets, between abc[0-3], can switch at any given point of time:

  ```
  set_mutex_condition {abc[0] abc[1] abc[2] abc[3]}
  ```

  Each of the specified attackers will be evaluated and the worst one will be used.

set_noise_lib_pin

```
set_noise_lib_pin
[-help]
-from string
-to string
```

Allows you to map noise properties of an instance pin to that of different library cell pin.

Parameters

`-help`	Prints out the command usage.
`-from` *string*	Specifies the library pin name whose definition will be mapped from the given pin name.
`-to` *string*	Specifies the library pin name whose definition will be mapped to the given pin name. When multiple `set_noise_lib_pin` commands are run on the same `-to` pin, the software will honor the first command only. **Note**: The `-from` and `-to` parameters also accept library cell pin names (that can be retrieved using the `get_lib_pins` command). These parameters do not honor collections or instance pin names.

Examples

- The input pin A of the `macro_cell` cell inherits the noise characteristics of the input pin I of the library cell, `ccs_noise_lib/inv`. This information is used to calculate the noise immunity of the pin, `library1/macro_cell/A`.
  ```
  > set_noise_lib_pin -to library1/macro_cell/A -from ccs_noise_lib/inv/I
  ```
- The following syntax specifies the library pin name whose definition will be mapped from the given pin name to the given pin name:
  ```
  > set_noise_lib_pin -from [from_lib/from_cell/from_pin] -to [to_lib/to_cell/to_pin]
  ```
- The following syntax specifies that a bidi pin is mapped to two from pins; that is, – 1 input pin and 1 output pin.
  ```
  > set_noise_lib_pin -to library1/macro_cell/PAD -from "ccs_noise_lib/inv/I
  ccs_noise_lib/inv/Z"
  ```

Related Information

- report_noise_lib_pin
- reset_noise_lib_pin
- get_lib_pins
- report_noise

set_outbound_report

```
set_outbound_report
[-help]
-delay {early | late | both}
[-out_file_prefix string]
[-report_constant_nets {true | false}]
[-type {lower | upper | both}]
```

Controls what extrapolation information is to be saved in the log file(s). The slew extrapolation information can be stored into separate files.

When this command is not specified, slew extrapolation will be reported for both minimum and maximum analysis, if outbound reporting has been enabled.

Parameters

`-help`	Prints out the command usage
`-delay {early | late | both}`	Control whether extrapolation information is to be shown for min or max (or both) analyses. *Default*: `both`
`-out_file_prefix string`	Specify the prefix to be used in the name(s) of the report file(s). *Default*: `design-name`
`-report_constant_nets {true | false}`	Controls whether extrapolation information is to be shown for constant nets or not.
`-type {lower | upper | both}`	Controls whether extrapolation information is to be shown for lower or upper (or both) outbound slews.

Examples

The following command will will dump slew extrapolation information for maximum analysis into the myreport.dc.outbound.slew.late.<view-name>.txt file:

```
> set_outbound_report -delay late -out_file_prefix myreport
```

set_quiet_aggressor

```
 set_quiet_aggressor
[-help]
[-analysis_type {early | late}]
[-aggressor string]
[-transition {rise | fall}]
[-victim string]
[-view string]
```

Specifies one or more coupled nets as non-attackers (silent) for the delay and glitch analysis of a given victim net.

You can run the `update_glitch` or `update_timing` command and view the results in the `report_noise` or `report_timing` command output.

Parameters

`-help`	Prints out the command usage.
`-analysis_type {early | late}`	Specifies the analysis type for which the attackers will be assumed quiet. If no value is specified, attackers are skipped for both `early/late` analysis.
`-aggressor string`	Specifies the attackers which should be made quiet for given victim net(s). If no attacker is specified, all the aggressors of given victim(s) will be ignored. **Note**: If no victim is specified, the specified attackers are made quiet for all victim nets in the design.
`-transition {rise | fall}`	Specifies the victim transition for which attackers will be assumed quiet. If no value is specified, attackers are skipped for both `rise` and `fall` transitions of the victim net.
`-victim string`	Specifies a list of victim nets for which attackers should be made quiet. Specific attackers can be skipped using the `-aggressor` option of this command. If the `-aggressor` option is not specified, all the attackers of the given nets are made silent and SI analysis will not be performed on these nets. **Note**: If this option is not specified, the attackers specified in the `-aggressor` option are made quiet for all the nets in the design.
`-view string`	Specifies the view for which attackers should be made quiet. If no view is specified, attackers are assumed quiet in all views

Examples

Consider a design having an attacker `agg` to victim nets `vic1` and `vic2`. The following command sets the attacker `agg` as quiet for both the victim nets `vic1` and `vic2`.

```
set_quiet_aggressor -aggressor {agg}
```

- The following command sets the attackers A1 and A2 as quiet when V1 is rising during late analysis.

  ```
  set_quiet_aggressor -victim V1 -aggressor A1 A2 -transition rise -analysis_type late
  ```

- The following command sets the attacker A1 quiet for victim V1 in the analysis view VIEW1. In this case, A1 may be a valid attacker during SI analysis for victim V1 in views other than VIEW1.

  ```
  set_quiet_aggressor -victim V1 -aggressor A1 -view {VIEW1}
  ```

Consider a design having victim `vic` with attackers `agg1` and `agg2`:

- The following command sets both the attackers `agg1` and `agg2` as quiet for the victim net `vic`.

  ```
  set_quiet_aggressor -victim {vic}
  ```

- The following command sets the attacker `agg1` as quiet for the given victim net `vic`

  ```
  set_quiet_aggressor -aggressor {agg1} -victim {vic}
  ```

Related Commands

- `report_noise`
- `report_timing`

si Category Attributes

The root attributes in the **si** category are:

- si_accumulated_small_aggressor_factor
- si_accumulated_small_aggressor_mode
- si_accumulated_small_aggressor_threshold
- si_aggressor_alignment
- si_clock_synchronicity
- si_delay_clock_delta_threshold
- si_delay_delta_threshold
- si_delay_delta_annotation_mode
- si_delay_enable_double_clocking_check
- si_delay_enable_logical_correlation
- si_delay_enable_report
- si_delay_separate_on_data
- si_enable_bus_attacker_correlation
- si_enable_glitch_overshoot_undershoot
- si_enable_two_stage_driver_weakening
- si_enable_drv_with_delta_slew
- si_enable_glitch_propagation
- si_enable_glitch_propagation_spice_deck
- si_enable_virtual_attacker_constituent_report
- si_glitch_accuracy_level
- si_glitch_analysis_type
- si_glitch_constrained_input_threshold_failure_point
- si_glitch_enable_dynamic_receiver_peak_limits
- si_glitch_enable_report
- si_glitch_input_threshold
- si_glitch_input_voltage_high_threshold
- si_glitch_input_voltage_low_threshold

- si_glitch_receiver_clock_peak_limit
- si_glitch_receiver_latch_peak_limit
- si_glitch_receiver_peak_limit
- si_individual_aggressor_clock_threshold
- si_individual_aggressor_simulation_filter
- si_individual_aggressor_threshold
- si_max_virtual_attacker_constituents
- si_nonlinear_aggressor_slew
- si_num_iteration
- si_reselection
- si_reselection_delay_threshold
- si_reselection_hold_slack
- si_reselection_setup_slack
- si_secondary_attacker_decoupling_factor
- si_skip_noise_model_check
- si_skip_timing_window
- si_switch_probability
- si_pessimistic_mode
- si_unconstrained_net_use_infinite_timing_window
- si_use_infinite_timing_window

To set the value of any root attribute, use the `set_db` command:

- `set_db` *`attribute_name value`*

To get the current value of any root attribute, use the `get_db` command:

- `get_db` *`attribute_name`*

To get the current value of all attributes in the `assign_pins` category, use the following command:

- `get_db -category si`

The SI attributes can be reported by using the following command:

- `get_db si_*`

si_accumulated_small_aggressor_factor

`si_accumulated_small_aggressor_factor` *`double`*

Default:1

Specifies the multiplication factor that controls the effect of the cap mode accumulated small attacker. An accumulated small attacker is an imaginary attacker that represents the sum of all the weak attackers on a specified net. To reduce the pessimism of using the accumulated small attacker, adjust this value to anything between 0 and 1. A value of 0 means that the cap mode accumulated small attacker will not be used.

Note: This option does not work with the current mode accumulated small attacker, which is default for process sizes 45nm and below.

si_accumulated_small_aggressor_mode

`si_accumulated_small_aggressor_mode {cap | current | zero_mean}`

Default: `current`

Specifies whether to use the current-matching based method for accumulated small attackers.

- `cap`: Uses the accumulated small attacker method, which matches the total coupling capacitance of an attacker with the small attackers that are being virtualized. This is the default mode for all process nodes above 45nm.
- `current`: Uses the current-matching based method for accumulated small attackers. With this method, the transitions on virtual attackers are modeled using Piece Wise Linear (PWL) waveforms. The PWL waveforms are generated in a way that the current induced by the transition matches the total current induced by all accumulated small attacker components.
- `zero_mean`: A new accumulation method where series of small attackers, that is, attackers less than the individual_attacker_threshold value, are assumed to transition in opposing directions and thus cancel one another in aggregate.

This is the default mode for 45nm and below process nodes.

si_accumulated_small_aggressor_threshold

`si_accumulated_small_aggressor_threshold` *`double`*

Default: 10.01

Changes the accumulated small attacker glitch threshold. If the virtual attacker produces a glitch whose peak is below this threshold, the component coupling capacitors of the accumulated small attacker are split into two ground capacitors and there is no accumulated small attacker for the victim. The value is specified as a ratio of VDD.
To disable the accumulated small attacker, set the value greater than 1.

si_aggressor_alignment

`si_aggressor_alignment {path | path_overlap | timing_aware_edge}`

Default: `path`

Specifies the attacker alignment.

- `path`: The victim worst arrival edge is used while determining the attackers. Crosstalk impact is computed according to the latest/earliest arrival edge of victim for max/min analysis.
- `path_overlap`: The full timing window of victim is considered while determining the attackers. Crosstalk impact is calculated according the victim edge, lying anywhere in timing window, which may experience maximum impact.
- `timing_aware_edge`: Realistic analysis similar to path-based analysis is performed. Each attacker's impact is annotated on victim transition where attacker timing window edge aligns with the victim. However, to ensure GBA bounds PBA analysis, the timing-aware edge mode handles some nets based on their timing properties in GBA analysis.

si_clock_synchronicity

Default: `synchronous`

Treats the clocks either synchronous or asynchronous to each other.

si_delay_clock_delta_threshold

`si_delay_clock_delta_threshold <double>`

Default: `-1`

Specifies the clock delta delay analysis threshold value.

si_delay_delta_threshold

`si_delay_delta_threshold <double>`

Default: `-1`

Specifies the delta delay analysis threshold value.

si_delay_delta_annotation_mode

`si_delay_delta_annotation_mode {lumpedOnNet | arc}`

Default: `arc`

Calculates the delay for each arc (cell arc and net arc) in a path. By default, AAE calculates the delay, including SI for each arc and if the user chooses to view the SI delay separately, it would show annotated to each arc. Alternatively, you can choose to annotate the SI delay solely to the net. In this case, the worst-case arc is used to calculated the SI delay and it is annotated solely to the net.

- `lumpedOnNet`: Lumps the delta delay on a the net, if it is broken out.
- `arc`: Separates the delta delay on each cell arc and net arc, as calculated.

si_delay_enable_double_clocking_check

`si_delay_enable_double_clocking_check`

Default: `false`

Enables double clocking check in AAE.
This option enables double clocking analysis and `report_double_clocking` command can be run to print a report.

si_delay_enable_logical_correlation

`si_delay_enable_logical_correlation`

Default: `false`

Enables the AAE-SI logical correlation. Logical correlation refers to correlation of logic between (aggressor) nets which are connected to each other. A simple case would be input and output nets of an invertor never switch in the same direction, so cannot be attackers to the same net simultaneously.

si_delay_enable_report

`si_delay_enable_report`

Default: `false`

Enables SI delay reports.

si_delay_separate_on_data

`si_delay_separate_on_data {true | false}`

Default: `false`

Separates incremental delays on signal nets.

si_enable_bus_attacker_correlation

`si_enable_bus_attacker_correlation {true | false}`

Default: `false`

Enables the software to treat bits of bus as correlated attackers.

When set to `true`, the bits of bus type nets are grouped together based on their cumulative weight.

The bus bits that act as attackers to a specific victim are treated as a group. All the attackers are grouped into different bundles where normal (non-bus) attackers form one group and bits of the same bus form multiple other groups. In case of a victim that has a number of small couplings to bits of a bus, the combined effect of a full bus can be used to decide whether an attacker is significant.

si_enable_glitch_overshoot_undershoot

```
si_enable_glitch_overshoot_undershoot {true | false}
```

Default: `false`

Enables overshoot/undershoot analysis and reporting.

The life span of transistors can be described by the overshoot and undershoot values. If a transistor is consistently exposed to a certain percentage of overshoot/undershoot, then it may suffer a reduced life span.

si_enable_two_stage_driver_weakening

```
si_enable_two_stage_driver_weakening {true | false}
```

Default: `false`

Enables the two-stage driver weakening mode.

A non-failing significant glitch at the input of a two-stage driver can propagate through the first stage and weaken the second stage. This might cause the crosstalk-induced glitch to amplify at the next stage receiver (though it may not be significant enough to propagate). When two-stage driver weakening is enabled, the software considers weakened holding resistance for the two-stage driver for glitch analysis.

The dual stage driver weakening honors the glitch propagation threshold criteria.

Note: This feature is supported with CCS-N models only.

si_enable_drv_with_delta_slew

```
si_enable_drv_with_delta_slew {true | false}
```

Default : `false`

When set to `true`, the `max_transition` violations are reported based on the SI slews. Also, this allows SI slew reporting/fixing by the `signoffOptDesign` command and should be applied prior running the `signoffTimeDesign` command.

si_enable_glitch_propagation

```
si_enable_glitch_propagation {true | false}
```

Default: `false`

Enables glitch propagation through single stage cells and driver weakening through single stage and dual stage cells. When this attribute is set to `true`, the software does not delete the glitch waveform saved on instance input pins. The waveform is later used by spice deck to propagate glitch at the next stage receiver pin.

si_enable_glitch_propagation_spice_deck

```
si_enable_glitch_propagation_spice_deck {true | false}
```

Default: `false`

Enables the driver input glitch waveform to be printed in the SPICE deck such that SPICE can also perform single stage glitch propagation. When this option is set to `true`, the software does not delete the glitch waveform saved on instance input pins. The waveform is later used by spice deck to propagate glitch at the next stage receiver pin.

si_enable_virtual_attacker_constituent_report

```
si_enable_virtual_attacker_constituent_report {true | false}
```

Default: `false`

Enables virtual attacker constituent reporting.

The software provides support for virtual attacker constituent reporting (when the bus attacker correlation feature is enabled using the `si_enable_bus_attacker_correlation` attribute.

The virtual attacker constituent reporting feature is useful for identifying significant buses and their bits - this information can be used for glitch fixing.

si_glitch_accuracy_level

```
si_glitch_accuracy_level <int>
```

Default: 0

Specifies the accuracy levels for glitch analysis.

si_glitch_analysis_type

```
si_glitch_analysis_type {source | full_propagation}
```

Default: `source`

Specifies the glitch analysis type - `source` or `full_propagation`.

si_glitch_constrained_input_threshold_failure_point

```
si_glitch_constrained_input_threshold_failure_point {input | both}
```

Default: `input`

Allows you to control the glitch failure criteria when a user specifies an object-specific RIP threshold.

When set to `input`, the software skips ROP analysis and performs RIP-based failure check on the receiver pins for cases where object-specific RIP threshold is set using the `set_glitch_threshold -failure_point input -net` (or `-pin`) command.

When set to `both`, the software performs both RIP and ROP failure checks, wherever possible. If user-specified RIP failure criteria is violated, then it will be flagged as a RIP violation. If RIP criteria passes and receiver noise models exist, then the ROP is computed and checked against the ROP failure threshold for failure determination.

For the following cases where the object-specific RIP thresholds are not specified, the attribute behavior remains unchanged:

- For receiver pins with noise models present, ROP-based failure check is performed. The RIP-based failure criteria is not checked at all.
- For receiver pins with noise models missing, the RIP-based failure criteria is checked for violations.

si_glitch_enable_dynamic_receiver_peak_limits

```
si_glitch_enable_dynamic_receiver_peak_limits {true | false}
```

Default: `false`

Enables automatic computation of ROP threshold value of two-stage cells, using the CCS-noise data on timing arcs.

When set to `true`, the software computes the noise threshold of the last stage from the table of DC currents.

si_glitch_enable_report

```
si_glitch_enable_report {true | false}
```

Default : `false`

Enables the storage of glitch information for further debug using the `report_noise` command.

When set to `false`, the `time_design` command will still report glitches in the timingReports/*.SI_Glitches.rpt file but the report_noise command will not be available for use.

When set to `true`, the `report_noise` command may be used and glitch information will be more aligned to software signoff values.

si_glitch_input_threshold

`si_glitch_input_threshold` *float*

Default : 0.4

Specifies the input glitch failure threshold value during AAE analysis.

si_glitch_input_voltage_high_threshold

`si_glitch_input_voltage_high_threshold` *float*

Default : 0.4

Specifies the input glitch vh failure threshold value during AAE analysis. The value is a fraction of the local VDD – if VDD is 0.9V and the threshold is 0.4, then the criteria for failure is 0.4 * 0.9=0.36V.

During timing window re-convergence, only certain nets are re-selected for glitch analysis in each iteration (if any of the above input glitch threshold parameters is specified). The criteria for glitch re-selection is based on the specified input glitch threshold value, if this value is smaller than 30% of VDD. If not, then all the glitches larger than 30% of VDD are selected for re-computation in each iteration.

Default : 0.4

Specifies the input glitch vh failure threshold value during AAE analysis. The value is the fraction of local VDD – if VDD is 0.9V and the threshold is 0.4, then the criteria for failure is 0.4 * 0.9=0.36V.

During timing window re-convergence, only certain nets are reselected for glitch analysis in each iteration (if any of the above input glitch threshold parameters is specified). The criteria for glitch reselection is based on the specified input glitch threshold value, if this value is smaller than 30% of VDD. If not, then all the glitches larger than 30% of VDD are selected for re-computation in each iteration.

si_glitch_input_voltage_low_threshold

`si_glitch_input_voltage_low_threshold` *float*

Default: 0.4

Specifies the input glitch vl lower threshold value during AAE analysis. The value is a fraction of the local VDD - if VDD is 0.9V and the threshold value is 0.4, then the criteria for failure is 0.4 * 0.9=0.36V.

During timing window re-convergence, only certain nets are re-selected for glitch analysis in each iteration (if any of the above input glitch threshold parameters is specified). The criteria for glitch re-selection is based on the specified input glitch threshold value, if this value is smaller than 30% of VDD. If not, then all the glitches larger than 30% of VDD are selected for re-computation in each iteration.

si_glitch_receiver_clock_peak_limit

`si_glitch_receiver_clock_peak_limit` *float*

Default : 0.1 (10 percent of VDD)

Defines the glitch check limit for latch inputs in receiver peak mode. This parameter is specified as a ratio of vdd.

si_glitch_receiver_latch_peak_limit

`si_glitch_receiver_latch_peak_limit` *`float`*

Default : 0.15

Defines the receiver peak/glitch peak limit at all nodes. The value specified for this parameter is a ratio of vdd.

si_glitch_receiver_peak_limit

`si_glitch_receiver_peak_limit` *`double`*

Default: 0.15

Defines the receiver peak/glitch peak limit.

si_individual_aggressor_clock_threshold

`si_individual_aggressor_clock_threshold` *`double`*

Default: 0.015

Defines the AAE virtual attacker clock threshold.

si_individual_aggressor_simulation_filter

`si_individual_aggressor_simulation_filter { true | false }`

Default : true

Uses electrical filtering to reduce pessimism due to small attackers being preserved when their simulated value is under the individual attacker threshold value.

Note: This option can only be used when the accumulated small attacker is turned off (`-accumulated_small_attacker_threshold` is `1.00` or larger).

si_individual_aggressor_threshold

`si_individual_aggressor_threshold double`

Default: 1.5% or 0.015

Sets the glitch tolerance for individual attacker nets on the victim net. The software estimates the noise contributed by an individual attacker on the victim. If the peak of this noise exceeds the value of `individual_attacker_threshold`, the attacker is significant and is individually reported in the constituents section of the report. Otherwise, this aggressor becomes eligible for the virtual attacker. If the virtual attacker

peak does not exceed the value of `accumulated_small_attacker_threshold`, the aggressor nets in the virtual attacker are grounded and there is no virtual attacker.

Note: Minimum allowed individual attacker to be kept is `0.1%` of vdd, which is equivalent to an `individual_attacker_threshold` of `0.001`.

si_max_virtual_attacker_constituents

`si_max_virtual_attacker_constituents` *integer*

Default: `5`

Specifies the maximum number of virtual attacker constituents to report.

si_nonlinear_aggressor_slew

`si_nonlinear_aggressor_slew {true | false}`

Default: `true`

Uses a non-linear attacker. If set to false, the software uses a linear attacker.

si_num_iteration

`si_num_iteration` *value*

Default : 2

Specifies the maximum number of timing window iterations that should be performed during SI delay analysis. Performing delay analysis results in changes to the timing windows, which can potentially change the victim/attacker net combinations. Timing window iterations enable the software to reach to a point where the delay changes are negligible.

Note: The minimum allowed timing window iterations is 2 for performing SI analysis.

si_reselection

`si_reselection {delta_delay | slack | xcap_ratio}`

Default: `slack`

Specifies the SI reselection criteria during timing window iterations.

- `delta_delay`: Reselects a net for calculation if the delta delay on the net is greater than the delay value.
- `slack`: Reselects nets that have failing slacks during timing window iterations.
- `xcap_ratio`: Specifies the xcap ratio.

si_reselection_delay_threshold

`si_reselection_delay_threshold` *`double`*

Default : 1e-11

Allows you to change the delay reselection criteria.

si_reselection_hold_slack

`si_reselection_hold_slack` *`double`*

Default : -1e-12 seconds

Specifies the slack reselection threshold for hold.

si_reselection_setup_slack

`si_reselection_setup_slack` *`double`*

Default: -1e-12 seconds

Specifies the slack reselection threshold for setup.

si_secondary_attacker_decoupling_factor

`si_secondary_attacker_decoupling_factor` *`double`*

Default : 1

Specifies decoupling factor for secondary aggressors.

si_skip_noise_model_check

`si_skip_noise_model_check` *`string`*

Default : " "

Specifies the list of pins for which the noise model check will not take place.

si_skip_timing_window

`si_skip_timing_window` *`nets`*

Default : " "

Specifies the list of nets for which timing window is to be skipped. The timing window becomes infinite for the net when it is both a victim and aggressor.

When specified, only the last setting is retained. The settings prior to the previous setting are ignored.

si_switch_probability

`si_switch_probability` *`double`*

Default : 0.3

Sets the attacker switching probability. If you specify a value less than 0.0 or more than 1.0 , the switching probability is automatically set to the default value of 0.3.

si_pessimistic_mode

`si_pessimistic_mode {advanced | advanced_node | reduced_pessimism | increased_pessimism | high_pessimism}`

Default: `advanced`

Specifies the pessimistic mode setting for adjusting delay calculation results. You can specify one of the following options:

- `advanced`: This is the default mode. This option may give good results at 28nm, but for newer nodes other values may be more optimal.
- `advanced_node`: This option works for newer technology nodes, such as 7nm onwards. This mode can help improve alignment to external signoff analysis tool by adjusting pin capacitances, which may help with slew alignment, whilst maintaining high pessimism for delay calculation.
- `reduced_pessimism`: In cases where Innovus is too pessimistic when correlating to external signoff analysis tool, this mode can help reduce pessimism for a better match. This mode may be useful at technology nodes, such as 16nm.
- `increased_pessimism`: In cases where Innovus is too optimistic when correlating to external signoff analysis tool, this mode can help increase pessimism for a better match. This is an older option, `high_pessimism` or `advanced_node` option may give better results at newer nodes, such as 7nm.
- `high_pessimism`: In cases where Innovus is too optimistic when correlating to external signoff analysis tool, this mode can help increase pessimism for a better match. This mode is similar to advanced_node, but without the pin capacitance adjustments.

Note: This attribute impacts delay calculation only when the `design_pessimistic_mode true` attribute is specified. The `advanced_node`, `high_pessimism`, and `increased_pessimism` options should give results that are on the pessimistic side, and `advanced_node` can be more optimal for newer nodes.

si_unconstrained_net_use_infinite_timing_window

`si_unconstrained_net_use_infinite_timing_window {true | false}`

Default: `true`

Specifies that unconstrained nets will have an infinite timing window. To turn this off and use a constrained timing window, set this to `false`.

si_use_infinite_timing_window

```
si_use_infinite_timing_window {true | false}
```

Default: `false`

Uses infinite timing windows in a single SI run.

update_glitch

```
update_glitch
[-help]
[-effort {medium | high}]
```

Allows you to run glitch analysis without running SI delay analysis. This command provides the ability to run glitch analysis on all the nets in a design or on a subset of nets.

Using the `update_glitch` command without `report_timing` or `update_timing` will only run glitch analysis.

Note: It is not recommended to run the `update_glitch` command after the `report_timing` or `update_timing` command.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `update_glitch` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man update_glitch`.
`-effort {medium | high}`	Specifies the effort level for update glitch. When set to `high`, glitch analysis uses more accurate modeling of glitch for small attackers, thereby reducing pessimism. **Note**: The run time in `high` mode may be slightly higher, and is largely dependent on the distribution of coupling. *Default*: `medium`

Examples

- The following command will run glitch analysis without SI delay:
  ```
  > update_glitch
  ```

Related Commands

- report_noise
- report_timing
- update_timing_condition

write_spice_deck

```
write_spice_deck
[-help]
[-add_side_load_measurement {true | false}]
[-add_voltage_for_missing_sensitization]
[-bbox_cells string]
[-clock_mesh_depth integer]
[-clock_mesh_from_fall_pin string]
[-clock_mesh_from_pin string]
[-clock_mesh_from_rise_pin string]
[-clock_mesh_no_hard_stop_at_to_pins {true | false}]
[-clock_mesh_to_pin string]
[-enable_fast_simulation {true | false}]
[-enable_spectre_x {true | false}]
[-exclude_side_load_cells string]
[-exclude_side_load_cells_for_missing_subckt {true | false}]
[-force_delete_output_dir]
[-glitch_receiver_pin string]
[-ground_nets string]
[-include_cell_variation {true | false}]
[-lsf_job_string string]
[-measure_point {all | path | stage}]
[-merge_spice_deck {true | false}]
[-monte_carlo_sim_init_args string]
[-monte_carlo_sim_transient_args string]
[-out_dir string]
[-out_file_prefix string]
[-power_nets string]
[-report_fields string]
[-report_noise_cmd string]
[-report_timing_cmd string]
[-run_simulation]
[-segment {data_path | launch_path | launch_clock_path | capture_path all}]
[-segment_end string]
[-segment_start string]
[-si_sweep_range float]
[-si_sweep_steps integer]
[-si_sweep_victim {true | false}]
[-side_path_levels integer]
[-spectre_cmd_options string]
[-spectre_exe_path string]
[-spectre_analysis_options <string>]
[-spice_include_file string]
[-spice_model_files string]
[-spice_stimuli_file string]
[-spice_subckt_files string]
```

```
[-supply_voltage_file string]
[-temperature float]
[-through string]
[-use_control_param {true | false}]
[-use_distributed_load_for_side_receivers {true | false}]
[-user_defined_measure_statements {append | overwrite}]
[-user_defined_measure_statements_file string]
[-view string]
[-waveform_aware_pulse_width_checks {true | false}]
[-xtalk {delay | voltage_low_glitch | voltage_high_glitch}]
[ [ > | >> ] ]
```

Generates the SPICE trace for a path.

While generating a Spice Deck, the `write_spice_deck` command looks for library cells required to run Spice and adds all cells as ".include" in the generated Spice netlist.

The output SPICE deck includes:

- Active and passive devices such as field-effect transistors (FETs), capacitors, and resistors.
- Initial conditions and voltage sources
- Actual names of the gates, nets, and cell instances, which helps the simulator interpret this information effectively. For example, consider the following excerpt, which shows an entry for a gate named `U0` in the SPICE deck:

```
*Gate U0
*--------------------
*Cell ports in order : x a VSS VDD
*--------------------
XU0 U0/x U0/a 0 U0/VDD inv_ahvt_64
*--------------------
```

Note: By default, the software uses the default list of power node names (`VDD` and `VCC`) and ground node names (`GND` and `VSS`).

Note: In order to output a Spice deck that generates the same report output as the `report_timing` command, you must run the `write_spice_deck` command with the `-report_timing`, `-segment_start`, and `-segment_end` parameters.

Parameters

`-help`	Prints out the command usage.

`-add_side_load_measurement {true | false}`	Adds the side load measurement in clock mesh Spice deck. By default, Innovus only measures the latencies, delays, and slews of arcs that are part of clock mesh specified using the `-from` parameter. When set to true, the software additionally measures the latencies and slews of the side recievers at each stage. *Default*: `false`
`-add_voltage_for_missing_sensitization`	Adds the voltage for missing sensitization.
`-bbox_cells` *string*	Specifies a list of cells to be considered as black boxes.
`-clock_mesh_depth` *integer*	Creates clock mesh spice deck of depth.
`-clock_mesh_from_fall_pin` *string*	Creates the spice deck with mesh net starting from falling transition of specified from pin.
`-clock_mesh_from_pin` *string*	Createsthe spice deck with mesh net starting from pin.
`-clock_mesh_from_rise_pin` *string*	Creates the spice deck with mesh net starting from rising transition of specified from pin.
`-clock_mesh_no_hard_stop_at_to_pins {true | false}`	When it is `true`, it will traverse down all the way till it hits the clock pin of a flip-flop or or if it passes the depth specified with the `-depth` parameter.
`-clock_mesh_to_pin` *string*	Creates the spice deck with mesh net ending at to pins.
`-enable_fast_simulation {true | false}`	When set to `true`, the software enables the fast simulation mode. In this mode, additional clock cycles are not required to initialize sequential cells in the Spice deck - this speeds up Spice simulations. All the sequential cells are initialized using the initial conditions and only a single clock edge is applied as input. By default, this parameter is set to `false`. In this mode, two clock cycles are applied as input to the clock network, where the first triggering edge is used for initialization of sequential cells, and the second edge is used for delay measurement.
`-enable_spectre_x {true | false}`	Enables the spectreX license checkout when launching Spectre.
`-exclude_side_load_cells` *string*	Specifies a list of cells for which the pin capacitance will be used for side loads instead of instantiating them.

`-exclude_side_load_cells_for_missing_subckt {true | false}`	When set to `true`, the software will add side load pin capacitance for missing Spice subckt cell instead of instantiating it.
`-force_delete_output_dir`	Removes the directory specified by the user using the `-out_dir` parameter.
`-generate_output_only {true | false}`	Generates the report output only, without triggering any simulations. *Default*: `false`
`-glitch_receiver_pin` *string*	Specifies receiver pin for glitch spice deck.
`-ground_nets` *string*	Specifies a list of ground nodes.
`-include_cell_variation {true | false}`	Writes out the Spice deck required to run simulations. When set to `true`, the software runs Monte-Carlo simulations from within the Innovus software. *Default*: `false`
`-lsf_job_string` *string*	Specifies a handle to the farm job for launching the simulation.
`-monte_carlo_sim_init_args` *string*	Passes necessary Monte Carlo parameters that are required to run MC simulations. The Monte Carlo analysis refers to “statistics blocks”, where statistical distributions and correlations of netlist parameters are specified. This parameter initializes these parameters under the statistics block. **Note**: This parameter can be used when `-include_cell_variation` is set to `true`.
`-monte_carlo_sim_transient_args` *string*	Initializes full transient time of simulation and transient time step.
`-measure_point {all | path | stage}`	Specifies whether to add all, path, or stage measurements. *Default*: `all`
`-merge_spice_deck {true | false}`	Allows you to enable merged Spice deck.
`-out_dir` *string*	Specifies the name of the output directory that will contain the generated delay and/or glitch spice decks. If simulation is also performed using the `-run_simulation` parameter, the Spice simulations output will also be saved in this directory.

`-out_file_prefix` *string*	Adds the specified prefix to the generated Spice deck files. If this parameter is not used, one of the following default names of the generated Spice decks will be used: • `path_<index>_setup.sp: for setup paths` • `path_<index>_hold.sp: for hold paths` • `path_<index>_vl_glitch.sp: for VL glitch` • `path_<index>_vh_glitch.sp: for VH glitch` The `<index>` in the above file names is controlled based on the number of decks that are generated using the `write_spice_deck` command. A user-specified string (specified using this parameter) is used as a prefix to the above mentioned file names.
`-power_nets` *string*	Specifies list of power nodes.
`-report_fields` *string*	Changes run path simulation internal reporting format. The format columns should be compatible with those in the `report_timing_format` attribute. These settings will be used for the subsequent `write_spice_deck` command statement.
`-report_noise_cmd` *string*	Specifies the `report_noise` command parameters as a list of arguments within brackets. When specified, the Spice deck will be generated for all the receiver pins that are reported by the `report_noise` command for the specified options. When specified with the `-run_simulation` parameter, the software writes out a comparison report comparing various glitch parameters between Tempus and Spice, such as, RIP, input glitch area, and ROP (only if ROP node name is available in the noise models).
`-report_timing_cmd` *string*	Specifies arguments to `report_timing` command.
`-run_simulation`	Enables path simulation in addition to generating Spice Deck.

`-segment {data_path | launch_path launch_clock_path | capture_path all}`	Creates the Spice deck of a segment out of the path spice deck. The following command/global variable settings have the following impact on the `-segment` parameter options: • `-path_type full_clock` is defined under the write_spice_deck-report_timing parameter, then the -segment default value is `launch_path`. • `-path_type full_clock` is not specified, then default value is `data_path`. • `timing_disable_retime_clock_path_slew_propagation` attribute is set to `false`, the default value is `launch_path` (irrespective of `-path_type` settings). • `timing_disable_retime_clock_path_slew_propagation` attribute is set to `true` (default), then `launch_clock_path` and `capture_path` options are supported only when the `-path_type full_clock` option of the `write_spice_deck -report_timing` parameter is used.
`-segment_end` *`string`*	Specifies the name of the segment end term.
`-segment_start` *`string`*	Specifies the name of the segment start term.
`-si_sweep_range` *`float`*	Range to sweep the victim.
`-si_sweep_steps` *`integer`*	Number of steps by which the victim is swept.
`-si_sweep_victim {true | false}`	Enables victim sweep.
`-side_path_levels` *`integer`*	Specifies side path level.
`-spectre_cmd_options` *`string`*	Specifies command line options for Spectre.
`-spectre_exe_path` *`string`*	Specifies location of Spectre path.
`-spectre_analysis_options <string>`	Adds Spectre analysis options to the run. When specified, the software provides instructions to Spectre, e.g., run a session with extra accuracy.
`-spice_include_file` *`string`*	Specifies the file to be included in the generated spice deck.
`-spice_model_files` *`string`*	Specifies the model file, to be included in generated spice.

`-spice_stimuli_file string`

Allows you to specify the file path containing stimuli to be used for the specified library cells. The cells' sensitization coming from the user-specified stimuli file takes precedence over Tempus inferred sensitization.

Syntax for delay Spice decks is given below:

```
lib_cell string <from | from_rise | from_fall>
string <to | to_rise | to_fall> string input_vector
string num_clk_pulse integer
```

Syntax for glitch Spice decks is given below:

```
lib_cell string glitch_type <VL | VH> output_pin
string input_vector string num_clk_pulse integer #
For victim driver sensitization
lib_cell cell_name glitch_type <VL | VH> input_pin
string input_vector string num_clk_pulse integer #
For victim receiver sensitization
```

The `num_clock_pulse integer` holds significance in case of sequential arcs, and specifies where to align the data pin transition with respect to the clock transition.

For example, a value of ‘1’ (default setting) specifies that the data pin transition needs to be aligned before the last clock transition where delay measurement is intended. This value can be used in case of regular flip-flops.
The value ‘2’ specifies that data pin transition needs to be aligned with respect to the second-last triggering clock edge. This value can be used for cascaded 2-bit flip flops.

Additionally for sequential arcs, the first pin in the user-specified input vector needs to be the data pin, the signal of which will be latched in the sequential cell.

Here are some examples of stimuli file:

- The following example shows the stimuli file containing sensitization for CP(rise)->Q(fall) and CP(rise)->Q(rise) timing arcs of library cell FF1:

  ```
  lib_cell FF1 from_rise CP to_fall Q input_vector
  !D&CDN&!SE&!SI num_clk_pulse 2
  lib_cell FF1 from_rise CP to_rise Q input_vector
  D&CDN&!SE&!SI num_clk_pulse 2
  ```

- The following example shows the stimuli file containing sensitization for the victim driver cell AOI for both VL and VH glitches:

  ```
  lib_cell AOI glitch_type VL output_pin Z
  ```

	`input_vector !A&!B&!C` `lib_cell AOI glitch_type VH output_pin Z`
`-spice_subckt_files` *string*	Specifies the subckt file, to be included in generated spice.
`-supply_voltage_file` *string*	Specifies a user-defined power/ground voltage file.
`-through` *string*	Creates the Spice deck with mesh net going by `-through` points.
`-temperature` *float*	Specifies a user-defined temperature to be used in SPICE simulations. *Unit*: Celsius
`-use_control_param {true | false}`	Allows you to control the power/ground voltage levels in the created Spice deck through user-specified parameters. When set to `true`, an additional file with the same name as the deck, and extension .control is created that contains the parameterized design voltage levels of the PG pins of all the cells instantiated in the deck. Your can modify the *.control file as per the requirement without modifying the original deck.
`-use_distributed_load_for_side_receivers {true | false}`	Allows you to control whether to use the receiver output load as distributed or lumped load. *Default*: `false`
`-user_defined_measure_statements {append | overwrite}`	Specifies whether to overwrite the default measurement statements in the Spice deck by user-defined measurements specified using the `-user_defined_measure_statements_file` parameter, or append it with default measure statements. *Default*: `append`

`-user_defined_measure_statements_file` *string*	Allows you to add the delay/slew measurement statements for any given segment of the path for which Spice deck is generated. The format of the file is given below: `<MEASURE_TAG_NAME TRIGGER_PIN_NAME TARGET_PIN_NAME>` Multiple such lines with unique tags can be specified in the file. The measure statement is written in the Spice deck for delay measurement between `TRIGGER_PIN` and `TARGET_PIN`, and slew measurement at `TARGET_PIN`.
`-view` *string*	Specifies the view name, if glitch spice deck is create for other than default view.
`-waveform_aware_pulse_width_checks {true | false}`	Generates the Spice deck for waveform-aware pulse width checks.
`-xtalk {delay | voltage_low_glitch | voltage_high_glitch}`	Generates the SI SPICE Deck for the stage. A stage can start with a victim driver input to victim receiver input or receiver output. You can specify one of the following options: • `delay:` Used to dump the SI delay SPICE deck. • `voltage_low_glitch` and `voltage_high_glitch`: Specifies the type of glitch (VL or VH) the SPICE deck will be used for. This option must be used for glitch SPICE deck generation.
`>` *filename*	Redirects the output to a file.
`>>` *filename*	Appends to an output file.

Examples

- The following section in SPICE deck specifies global supplies and transient analysis window time.

```
*--------------------
*Command and Options
*--------------------
.GLOBAL vdd vbbnw
V0Xvdd  vdd 0 1.0
V0Xvbbnw vbbnw 0 1.0
.GLOBAL vss vbbpw
V0Xvss vss 0 0
V0Xvbbpw vbbpw 0 0
.TEMP -40.0000
.TRAN 1p 10256p
*--------------------
```

- The following section shows an example of a net and its instance connections.

```
*-------------------------
*Net Instances
*-------------------------
*Net module1/inst_1/net
*-------------------------
Xmodule1/inst_1/net module1/inst_1/q
+ module1/inst_2/a wc:module1/inst_1/q
*-------------------------
```

- The following section shows an example of a gate instance and its port connections.

```
*---------------------------------------
*Gate Instances
*---------------------------------------
*Gate module1/inst_2
*---------------------------------------
*Cell ports in order : a y vss vdd vbbpw vbbnw
*---------------------------------------
Xmodule1/inst_2/a module1/inst_2/y
+ 0 module1/inst_2/vdd 0 module1/inst_2/vbbnw
+ inv1
*---------------------------------------
```

- The following command run simulations and then generates a report:

```
write_spice_deck -report_timing "-net -from clk -to ca53/as_12/ff/D -path_type full_clock" -
```

```
run_simulation -include_cell_variation true > spice.socv.rpt
```

- The following command generates a report (if simulations have already been run):

```
write_spice_deck -generate_output_only true -report_timing "-net -from clk -to ca53/as_12/ff/D
-path_type full_clock" -run_simulation -include_cell_variation_true -format {arrival
spice_arrival spice_arrival_detail arrival_error_detail hpin} > spice.socv.rpt
```

- The following section shows the measurement statements for delay and slew.

```
*----------------------
*Measurement statements
*----------------------
.MEASURE TRAN Stage_0001_module1/inst_1/clk_DELAY_
module1/inst_1/q                                          ---------clk to q delay
+ TRIG V(module1/inst_1/clk) VAL=0.5 TD=167.9p RISE=1          of module1/inst_1
+ TARG V(module1/inst_1/q) VAL=0.5 TD=167.9p RISE=1
.MEASURE TRAN Stage_0001_module1/inst_1/q_SLEW
+ TRIG V(module1/inst_1/q) VAL=0.30 TD=167.9p RISE=1
+ TARG V(module1/inst_1/q) VAL=0.70 TD=167.9p RISE=1   ---------Slew at
.MEASURE TRAN Stage_0002_module1/inst_1/q_DELAY_                module1/inst_1/q pin
module1/inst_2/a
+ TRIG V(module1/inst_1/q) VAL=0.5 TD=167.9p RISE=1    ---------Interconnect Delay
+ TARG V(module1/inst_2/a) VAL=0.5 TD=167.9p RISE=1          and Input Slew
.MEASURE TRAN Stage_0002_module1/inst_2/a_SLEW
+ TRIG V(module1/inst_2/a) VAL=0.30 TD=167.9p RISE=1   ---------Interconnect Delay
+ TARG V(module1/inst_2/a) VAL=0.70 TD=167.9p RISE=1         and Input Slew
```

Note: The `DELAY` and `SLEW` keywords in the `MEASURE` statements indicate that these are the measurement statements for delay and slew respectively.

- The following command sets the simulation time stop criteria to be 13.40 ns and step time to be 1ps:

```
tempus > write_spice_deck -mc_simulation_init_args "stop=1.34e-08 step=1p"
```

- The following command writes out statistics block in a deck to run Monte Carlo simulations:

```
tempus > write_spice_deck -mc_simulation_init_args "numruns=3000 seed=1234 distribute=fork
numprocesses=8 variations=mismatch scalarfile=MCDataSetLoad.txt dut=[%s]"
simulator lang = spectre

mc1 montecarlo numruns=3000 seed=1234 distribute=fork numprocesses=8 variations=mismatch
scalarfile=MCDataSetLoad.txt dut=[ XStage2.XCTS_ccl_a_inv_01730 XStage3.XCTS_ccl_inv_01726
XStage4.XCTS_ccl_inv_01718 XStage5.XCTS_ccl_a_inv_01712 \
XStage6.XCTS_cpc_sk_inv_03405 XStage7.XCTS_cpc_sk_inv_03406 XStage8.XCTS_cdb_inv_03303
XStage9.XCTS_cdb_inv_02837 \
XStage10.XCTS_ccl_inv_01669 XStage11.XCTS_ccl_a_inv_01540 XStage12.XCTS_cpc_sk_inv_03860
XStage13.XCTS_cpc_sk_inv_03859 \
XStage14.Xugenq_uaesflushu1_aes_flush_u1_reg XStage15.XPLACEOPTFE_RC_149_0
```

```
XStage16.XPLACEOPTFE_RC_150_0 \
XStage17.XU22704 XStage18.XU22707 XStage19.XU22857 XStage20.XPLACEOPTFE_OFC2135_n30081
XStage21.XU23049 \
XStage22.XU140015 XStage23.XU132065 XStage24.XU132066 XStage25.XU132710 XStage26.XU132716
XStage27.XU133277 \
XStage28.XU133278 XStage29.XU133279 XStage30.XU133285 ]

{
tran1 tran stop=2.88312e-08 step=1p
}
simulator lang = spice
```

Related Commands

- report_timing

35

Structured Data Path Commands and Attributes

- add_sdp_rows_blockages
- check_sdps
- create_sdp
- delete_sdp_rows_blockages
- place_sdp Category Attributes
- place_sdps
- read_sdp_file
- set_sdp_origin
- update_sdp
- write_sdp_file

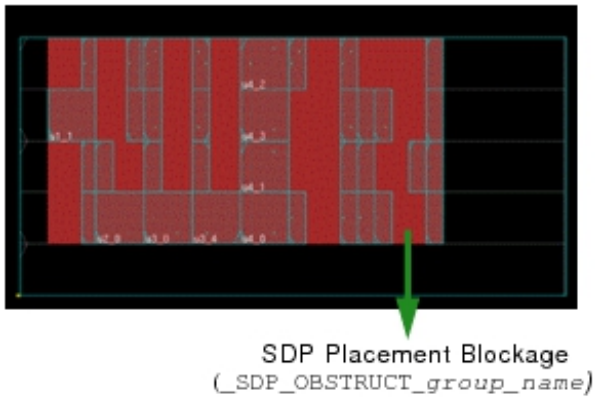
add_sdp_rows_blockages

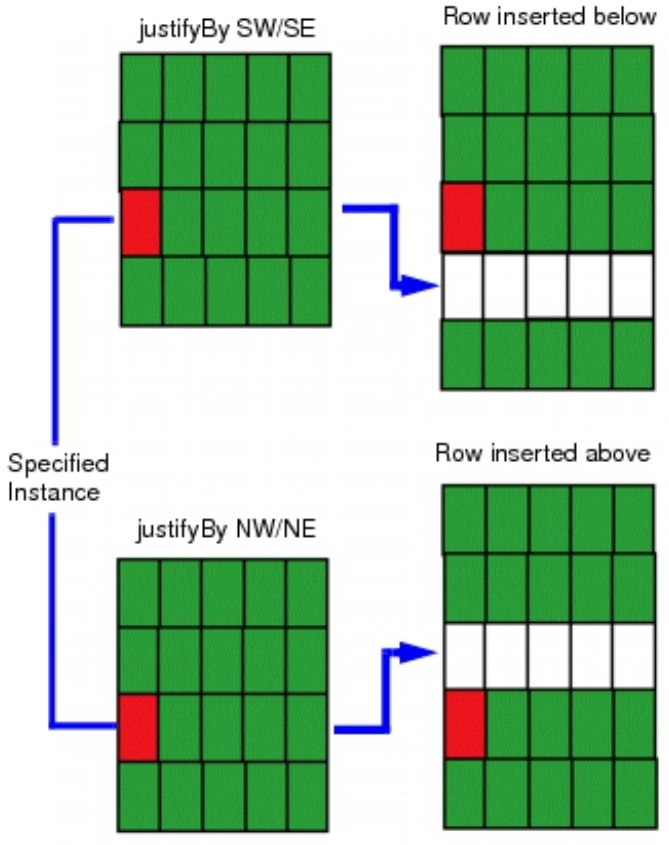
```
add_sdp_rows_blockages
[-help]
{[-blockages {group_name_list}]| [-empty_row instance_name [-num_row value]]}
```

Creates a placement blockage on top of the specified SDP group OR inserts one or more empty rows across the whole die above/below the specified instance.

Use the `add_sdp_rows_blockages` command to prevent cells from being placed inside the SDP group area. You can also use this command to create empty rows for clock or IPO buffers that will be inserted later on in the flow.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `add_sdp_rows_blockages` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man add_sdp_rows_blockages`.
`-blockages` *`group_name_list`*	Creates a placement blockage over the specified SDP group(s).  SDP Placement Blockage (_SDP_OBSTRUCT_*group_name*) *Data_type*: string, optional
`-empty_row` *`instance_name`*	Inserts empty row(s) above/below the specified instance based on the `justifyBy` constraint of the group to which the instance belongs. The empty row(s) are inserted across all SDP groups. The number of rows to be inserted is specified by the `-num_row` parameter. If `-num_row` is not specified, one empty row is inserted by default. **Note:** The core boundary will be extended to accommodate the newly added rows if the current core box is not big enough.



Data_type: string, optional

`-num_row` *value*	Indicates the number of empty rows to be inserted. *Data_type*: int, optional

Examples

- The following example creates a placement blockage over the SDP group `sdpGrp1`.
  `add_sdp_rows_blockages -blockages sdpGrp1`
- The following example inserts two empty rows below the `DTMF_INST/ROM_512x16_0_INST`

instance.

```
add_sdp_rows_blockages -empty_row DTMF_INST/ROM_512x16_0_INST -num_row 2
```

Related Information

- `create_sdp`
- `delete_sdp_rows_blockages`
- `update_sdp`
- sdp object in *Stylus Common UI Database Object Information*
- Using Structured Data Paths chapter of the *User Guide*
- Floorplan Menu chapter of the *Menu Reference*
  - *Structured Data Path*

check_sdps

```
check_sdps
[-help]
[-out_file fileName]
[-sdps {sdp_name...}]
```

Checks current SDP placement against the SDP constraints, such as order, alignment, and orientation, which may have been originally specified via an SDP relative placement file or a set of TCL commands and reports violations. The `check_sdps` command also creates markers in the GUI for any SDP violation it detects. The violation markers can be viewed in the Violation Browser.

Whenever you perform any manual editing and/or optimization step, you can use `check_sdps` to check for any resulting SDP violations. This ensures that current placement is still valid as per the SDP constraints you had originally specified.

Before checking SDP placement, you should have:

- Imported the design
- Created and placed SDP groups

Parameters

`-help`	Prints a brief description that includes type and default information for each `check_sdps` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man check_sdps`.
`-out_file` *`fileName`*	Generates a detailed report for the following checking categories: • SDP group/instance orientation • SDP group/instance justifyBy constraint • Alignment by pin name constraint • SDP group/instance flip constraint • Skip space constraint • SDP group/instance overlapping • Inside parent box violation For each of the above checking category, the report includes the following information: • Total number of violations • List of SDP group/instance names that have that violation *Default*: *`designName`*`.checkSdp.rpt` *Data_type*: string, optional
`-sdps {`*`sdp_name...`*`}`	Checks only the specified sdps. *Data_type*: string, optional

Example

- The following example checks SDP placement and generates the default report:

  ```
  check_sdps
  ```

  Sample output:

  ```
  ############################################################
  # Generated By: Cadence Innovus 18.10-d183_1
  # Generated on: Fri Dec 15 08:53:35 2017
  # HostName: noi-leenap
  # Design: DTMF_CHIP
  # Command: check_sdps
  ############################################################


  ############################################################
  Orientation and flip checking are only done for the object
  which constaint 'orient/flip' is specified in the sdp file.
  Due to the instance placement needs to match the row orientation,
  the instance actual orientation might not be same as the value
  that specified in the file. The orientation/flip checking will
  report this kind of mismatch
  ############################################################

  ERROR: check justifyBy fail, sdp group
  DTMF_INST/TDSP_CORE_INST/MPY_32_INST/M16X16_INST/i_361
  ERROR: check justifyBy fail, sdp group
  DTMF_INST/TDSP_CORE_INST/MPY_32_INST/M16X16_INST/i_375
  ERROR: check skipSpace fail, sdp group DPNULL

  Orientation violation: 0
  JustifyBy violation: 2
  AlignByPinName violation: 0
  SkipSpace violation: 1
  Overlap violation: 0
  InsideParentBox violation: 0
  ------------------------------------------------
  Total violation: 3
  ```

- The following example checks SDP placement and generates a detailed report named `checkSdp1.rpt`.

  ```
  check_sdps -out_file checkSdp1.rpt
  ```

Related Information

- `check_place`
- `create_sdp`
- `place_sdps`
- `update_sdp`
- `sdp` object in *Stylus Common UI Database Object Information*
- Using Structured Data Paths chapter of the *User Guide*
- Floorplan Menu chapter of the *Menu Reference*
    - *Structured Data Path*

create_sdp

```
create_sdp
[-help]
[-justify_by {SW|SE|NW|NE|N|W|S|E|MID}]
-name sdpGroupName
[-type {row|col|space}]
{{-skip_space value} | -skip_space_by_site_width value siteName | -skip_space_by_micron
value}
| {{{-insts WCardName... [-keep_order]} | -selected }
  [[-spread_group value ] | [-spread_group_by_site_width value siteName] | [-
spread_group_by_micron value]
| [[-num_row value] [-num_column  value]]]}
```

Creates an SDP group. The added object can be an instance or another SDP group.

The `create_sdp` command enables you to create a structured data path using TCL commands. You do not need to learn SDP relative placement syntax for creating a data path.

Before creating an SDP group, you should have:

- Imported the design

Parameters

`-help`	Outputs a brief description that includes type and default information for each `create_sdp` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man create_sdp`.
`-insts WCardName...`	Specifies one or more instance names. Use this parameter if you want to create the SDP group from the specified instances. You can additionally specify `-keep_order` if you want the tool to honor the order in which SDP instance are specified in the list. **Note:** Wildcards can be used to specify the instance names. *Data_type*: string, optional
`-justify_by {SW|SE|NW|NE|N|W|S|E|MID}`	

	Specifies the anchor point that will be used to align the SDP group you are creating. *Default*: `SW` *Data_type*: string, optional
`-keep_order`	Honors the order of instances in the SDP instance list that you specify as input with the `-insts` parameter. *Data_type*: bool, optional
`-name` *sdpGroupName*	Specifies the name of the SDP group you are creating. *Data_type*: string, required
`-num_column` *value*	Indicates the number of columns to be created. *Data_type*: int, optional
`-num_row` *value*	Indicates the number of rows to be created. **Note:** If you specify both `-num_row` and `-num_column`, use the `-type` parameter to specify the parent structure type (`row` or `col`). If you do not specify type, the default parent structure will be column. *Data_type*: int, optional

`-selected`	Indicates that you want to create the SDP group from the current set of selected instances. If the selected cells are already placed and you specify neither the `-numRow` or `-numCol` options nor the `-type` option, the command will automatically generate one column with rows where the SDP structure will be based on the physical location of the selected cells. The tool will print the following message to indicate that the SDP structure created will be derived from the physical location of the selected cells: `Command is running with physical mode extraction where SDP structure will be derived from the existing physical cell placement.` If the selected cells are not yet placed or if some of the selected cells are placed while others are not, the tool prints a message indicating that SDP physical mode extraction cannot be activated and SDP structure cannot be derived from the existing physical placement because the selected instance is not placed. *Data_type*: bool, optional
`-skip_space` *value*	Specifies a space value to be skipped. By default, if `-skip_space` is defined in a column, its value represents the number of skipped rows. If `-skip_space` is defined in a row, its value is for column skipping and represents the number of M2 tracks (pitch of first vertical layer). Skip space value can also be specified as number of widths of a specified row site by using the `-skip_space_by_site_width` *`value siteName`* parameter. However, row site width should be specified only for an SDP row, and not for an SDP column. Alternatively, you can specify the `-skip_space_by_micron` parameter to specify the distance to be skipped in microns (`um`). **Note:** Use the `-skip_space` parameter with the `-type space` option to add row/column spaces. *Data_type*: string, optional

`-skip_space_by_micron` *value*	Specifies the space value to be skipped in microns (`um`). If the specified distance value is not equal to the exact number of first vertical metal pitches or number of rows, `create_sdp` rounds off the value. **Note**: This option cannot be used with either `-skip_space` or `-skip_space_by_site_width`. *Data_type*: string, optional
`-skip_space_by_site_width` *value siteName*	Specifies the number of row/column space to be skipped and the name of the tech site. Uses the width of the specified site as the skip unit in the horizontal direction. **Note**: This option cannot be used with either `-skip_space` or `-skip_space_by_micron`. *Data_type*: list, optional
`-spread_group` *value*	Inserts space between group of instances. If spread_group is specified in a column, spacing will be the number of added rows. If spread_group is specified in a row, spacing will be the number of placement grids added between the columns in that row. *Data_type*: string, optional
`-spread_group_by_micron` *value*	Specifies the space to be inserted between groups of instances in microns (`um`). **Note**: This option cannot be used with either `-spread_group` or `-spread_group_by_site_width`. *Data_type*: string, optional
`-spread_group_by_site_width` *value siteName*	Specifies the number of row/column space to be inserted and the name of the tech site. Uses the width of the specified site as the spread unit in the horizontal direction. **Note**: This option cannot be used with either `-spread_group` or `-spread_group_by_micron`. *Data_type*: list, optional

`-type {row|col|space}`	Indicates whether the new group is a row, column, or an empty (skipped) space. *Default*: `col` *Data_type*: string, optional

Examples

- The following example creates an SDP group named `TestGrp` of type `space` with two spaces:
  ```
  create_sdp -name TestGrp -type space -skip_space 2
  ```
- The following example creates an SDP group named `sdpGrp1` of type `row` with four columns from the selected instances. The group is aligned in the South East direction:
  ```
  create_sdp -name sdpGrp1 -type row -justify_by SE -num_column 4 -selected
  ```
- The following example creates a row data path that has three columns, each with two rows, for all the instances that have name matched the pattern `DTMF_INST/TDSP_CORE_INST*`:
  ```
  create_sdp -name sdp_group1 -type row -insts DTMF_INST/TDSP_CORE_INST* -num_row 2
  -num_column 3
  ```
- The following command automatically generates one column with rows where the SDP structure will be based on the physical location of the selected cells, provided that the selected cells are already placed:
  ```
  create_sdp -selected -name group2
  ```

Related Information

- `add_sdp_rows_blockages`
- `delete_sdp_rows_blockages`
- `place_sdp` Category Attributes
- `place_sdps`
- `update_sdp`
- `sdp` object in *Stylus Common UI Database Object Information*
- Using Structured Data Paths chapter of the *User Guide*
- Floorplan Menu chapter of the *Menu Reference*
    - *Structured Data Path*

delete_sdp_rows_blockages

```
delete_sdp_rows_blockages
[-help]
{[-sdps {object_name_list} [-keep_cell_placement]] | -place_blockages
{sdp_group_name_list} | [-empty_row instance_name [-num_row value]]}
```

Deletes one of the following:

- SDP group(s) or element(s)
- A placement blockage associated with the specified SDP group(s)
- Empty row(s) adjacent to the specified instance

Use the `delete_sdp_rows_blockages` command to modify structure of existing data path(s) easily and to remove existing SDP groups or elements.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `delete_sdp_rows_blockages` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man delete_sdp_rows_blockages`.
`-empty_row` *instance_name*	
	Deletes the empty row(s) adjacent to the specified instance based on the `justifyBy` constraint of the group to which the instance belongs. If the group is justified by `SW`/`SE`, the empty rows below the specified instance are deleted. If the group is justified by `NW`/`NE`, the empty rows above the specified instance are deleted. **Note:** If the rows adjacent to the specified instances are not empty, an error message is returned. *Data_type*: string, optional

`-keep_cell_placement`	Preserves the cell placement of remaining SDP members when an SDP object is deleted. If this parameter is not specified, `deleteSdpObject` unplaces all SDP members while deleting the specified SDPs. This option can be used only with `-sdps`. *Data_type*: bool, optional
`-num_row` *value*	Indicates the number of empty rows to be deleted. **Note:** If this parameter is not specified, one empty row is deleted by default. *Data_type*: int, optional
`-place_blockages {`*sdp_group_name_list*`}`	
	Deletes the placement blockage over the specified SDP group(s). *Data_type*: string, optional
`-sdps {`*object_name_list*`}`	
	Deletes the specified SDP group or instance. If you want to delete more than one object, enclose the object names in {}. *Data_type*: string, optional

Examples

- The following command deletes the SDP group `TestGrp`.

  ```
  delete_sdp_rows_blockages -sdps TestGrp
  ```

- The following command deletes a placement blockage over the SDP group `sdpGrp1`.

  ```
  delete_sdp_rows_blockages -blockages sdpGrp1
  ```

- The following command deletes two empty rows below the `DTMF_INST/ ROM_512x16_0_INST` instance where the group to which `ROM_512x16_0_INST` belongs has `justifyBy` constraint set as `SW`.

  ```
  delete_sdp_rows_blockages -empty_row DTMF_INST/ROM_512x16_0_INST -num_row 2
  ```

Related Information

- `add_sdp_rows_blockages`
- `create_sdp`
- `update_sdp`
- `sdp` object in *Stylus Common UI Database Object Information*
- Using Structured Data Paths chapter of the *User Guide*
- Floorplan Menu chapter of the *Menu Reference*
    - *Structured Data Path*

place_sdp Category Attributes

The root attributes in the `place_sdp` category are:

- place_sdp_clock_location
- place_sdp_disable_extend_core
- place_sdp_honor_dont_use
- place_sdp_honor_orient
- place_sdp_legalization
- place_sdp_legalization_effort
- place_sdp_max_move_distance
- place_sdp_num_column
- place_sdp_place_report
- place_sdp_pre_fixed_cells_blockage_direction

To set the value of any root attribute, use the `set_db` command:

```
set_db attribute_name value
```

To get the current value of any root attribute, use the `get_db` command:

```
get_db attribute_name
```

To get the current value of all attributes in the `place_sdp` category, use the following command:

```
get_db -category place_sdp
```

place_sdp_clock_location

```
place_sdp_clock_location {bottom | center}
```

Default: `center`

Data_type: string, read/write

Specifies where the Integrated Clock Gating (ICG) cell will be placed in an SDP register column.

Related Commands

- `place_sdps`

place_sdp_disable_extend_core

`place_sdp_disable_extend_core {true | false}`

Default: `true`

Data_type: bool, read/write

Disables extension of the core boundary. If you set `place_sdp_disable_extend_core` to `true` before running the `place_sdps` command, the software does not adjust the core boundary to accommodate all SDP placements. Instead, the software places the SDP elements outside the core boundary if they do not fit within the boundary.

Example

- The following command disables the extension of the core boundary if the boundary cannot accommodate all SDP placements:

  ```
  set_db place_sdp_disable_extend_core true
  ```

Related Commands

- `place_sdps`

place_sdp_honor_dont_use

`place_sdp_honor_dont_use {true | false}`

Default: `false`

Data_type: bool, read/write

Specifies whether the `dont_use` attribute on base cells should be honored by the `place_sdps` command during EEQ swapping in SDP legalization. By default, `place_sdps` takes all cells into account. If `place_sdp_honor_dont_use` is set to `true`, `place_sdps` ignores the cells with the `dont_use` attribute.

Example

- Use the following command if you want the SDP placer to ignore the cells with the `dont_use` attribute:

  ```
  set_db place_sdp_honor_dont_use true
  ```

Related Commands

- `place_sdps`

place_sdp_honor_orient

```
place_sdp_honor_orient {true | false}
```

Default: `false`

Data_type: bool, read/write

Honors user-specified orientation during legalization.

User-specified orientation can be set using the `set_db` command. Note that the SDP writer writes out orientation information only if its value is defined by the user through an SDP file or a TCL command.

Example

- Use the following command if you want the SDP placer to honor user-specified orientation:

  ```
  set_db place_sdp_honor_orient true
  ```

Related Commands

- `place_sdps`

place_sdp_legalization

```
place_sdp_legalization {NONE SW SE NW NE AUTO}
```

Default: `NONE`

Data_type: enum {NONE SW SE NW NE AUTO}, read/write

Specifies the corner of the SDP box that will be kept fixed while legalizing an SDP instance.

Related Commands

- `place_sdps`

place_sdp_legalization_effort

`place_sdp_legalization_effort {low medium high}`

Default: `low`

Data_type: enum {low medium high}, read/write

Specifies the legalization effort in placing SDP groups.

When you run `place_sdps`, the SDP groups are legalized on the basis of their order, which is determined by their size. Bigger SDP groups are placed first, and multiple iterations are used to place and align the remaining SDP groups. You can also control the legalization effort applied by using the `place_sdp_legalization_effort` attribute. The higher the setting of this attribute, the more the number of iterations attempted to try and place the SDP groups legally.

Related Commands

- `place_sdps`

place_sdp_max_move_distance

`place_sdp_max_move_distance` *`value`*

Default: `0.0`

Data_type: double, read/write

Specifies how far an SDP can be moved from its original clustering location to resolve overlaps. By default, this value is half the perimeter of the core box. The Manhattan distance between the new and original locations of the SDP must be less than the value of the `place_sdp_max_move_distance` attribute.

The `place_sdp_max_move_distance` attribute is honored by the `place_detail` command. SDP columns start at the center-of-mass of the instances and then try to find a legal location without overlaps. If a legal location cannot be found within *`value`* from the starting point, the

`place_sdp_max_move_action` attribute is used to determine what to do.

Example

- The following set of commands control SDP movement from its original cluster location to resolve overlaps. SDP groups are deleted if their overlap cannot be resolved:

```
set_db place_sdp_max_move_distance 30
set_db place_sdp_max_move_action delete_sdp_group
place_design
```

Related Commands

- `place_design`
- `place_sdps`

place_sdp_num_column

`place_sdp_num_column` *`numberOfCol`*

Default: `2`

Data_type: int, read/write

Specifies the number of flip flop columns for each SDP group.

Related Commands

- `place_sdps`

place_sdp_place_report

`place_sdp_place_report` *`file_name`*

Default: `""`

Data_type: string, read/write

Generates a detailed SDP placement report when `place_sdps` is run.

Default: If `place_sdp_place_report` has not been specified before running `place_sdps`, this report is

not generated.

Note: To disable the `place_sdp_place_report` attribute once it has been specified, specify *`file_name`* as a null string (`""`). The detailed SDP placement report will not be generated subsequently.

Example

- The following commands place elements or instances of all defined SDP groups and generate a detailed SDP placement report named `dtmf_chip.sdp.rpt`:

  ```
  set_db place_sdp_place_report dtmf_chip.sdp.rpt
  place_sdps
  ```

Related Commands

- `place_sdps`

place_sdp_pre_fixed_cells_blockage_direction

`place_sdp_pre_fixed_cells_blockage_direction {NONE X Y}`

Default: `NONE`

Data_type: enum* {NONE X Y}, read/write

Creates temporary placement blockage internally over pre-fixed cells (that is, well tap or power switch cells) in `X` or `Y` direction to prevent SDP cells to be placed in the horizontal or vertical empty spaces when resolving overlapping.

Related Commands

- `place_sdps`

place_sdps

```
place_sdps
[-help]
[-create_inst_group]
[-delete_overlap_cells {cell...} {sdp name...}]
[-sdps {group_name_list}]
[-no_fix_overlap | [-keep_sdp_area]]
```

Places elements or instances of all defined SDP groups in row(s) and column(s) based on user-defined relative placement information. The `place_sdps` command:

- Adjusts the core boundary automatically if the existing core is not big enough for placing all specified SDPs
- Resolves automatically the overlaps on the row/column cell list specified with attributes in the `place_sdp` category.
- Honors the `alignByPinName` SDP constraint for column groups
- Marks all placed SDP instances to fixed status
- Places the SDP boundary outside the die area on the right hand side if origin information is not defined.

Before placing SDP groups, you should have:

- Imported the design
- Defined SDPs using TCL commands or modified SDPs, as required

Parameters

`-help`	Outputs a brief description of the `place_sdps` command. For a detailed description of the command, use the `man` command: `man place_sdps`.
`-create_inst_group`	Converts failed top sdp groups to instance groups. In the 2G flow, the tool supports the conversion of failed top SDP groups to instGroups by using this parameter. This option works only on top SDPs. This means the tool will convert the top SDPs to instGroups, irrespective of whether it was the top SDPs or child SDPs that failed. The instances in the instance groups are soft_fixed.
`-delete_overlap_cells {`*`cell...`*`} {`*`sdp name...`*`}`	Deletes specified physical cells that overlap with all or specified SDP groups. **Note:** If you do not specify *`sdp name ...`*, the specified physical cells are removed from all SDPs. *Data_type*: string, optional
`-keep_sdp_area`	Keeps the original SDP structure when resolving overlaps with other instances. Use this option if you want to resolve the cell overlap by moving the SDP structure instead of changing it. *Data_type*: bool, optional
`-no_fix_overlap`	Ignores overlaps with column/row cells that are specified using attributes in the `place_sdp` category. Use this option to take a look at how the groups will be placed before resolving overlaps. *Data_type*: bool, optional
`-sdps {`*`group_name_list`*`}`	Places only the specified SDP groups. *Data_type*: string, optional

Examples

- The following command places all defined SDP groups without removing overlaps:
  ```
  place_sdps -no_fix_overlap
  ```

Related Information

- `check_sdps`
- `create_sdp`
- `place_sdp` Category Attributes
- `update_sdp`
- `sdp` object in *Stylus Common UI Database Object Information*
- Using Structured Data Paths chapter of the *User Guide*
- Floorplan Menu chapter of the *Menu Reference*
  - *Structured Data Path*

read_sdp_file

```
read_sdp_file
[-help]
sdp_file file_name
[-hier_path path]
[-left_over_group]
[-origin llx lly]
[-skip_placement]
```

Reads in a relative placement file (in .sdp format) and then calls `place_sdps` to place all the structured data path (SDP) elements defined in the file. If `-skip_placement` is specified, placement is skipped.

If the SDP file refers to an instance that is not present in the netlist or design, the software issues a warning but continues to load the rest of the SDP file.

Before using `read_sdp_file`, you should have:

- Imported the design

Parameters

`-help`	Outputs a brief description that includes type and default information for each `read_sdp_file` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man read_sdp_file`.
`-hier_path` *`path`*	Specifies the hierarchical path name of the SDP elements specified in the SDP file. This path name is appended to all SDP instances defined in the file. *Data_type*: string, optional
`-left_over_group`	Stores all remaining instances in the current hierarchical path that do not belong to any SDP group into a new SDP group called *`SDPname`*`_LEFTOVER_RPLGRP`. An error is issued if a logical hierarchy has more than one SDP group. *Data_type*: bool, optional
`-origin` *`llx lly`*	Defines the SDP location by specifying the SDP block origin. The origin information is applied to all SDPs specified in the SDP file. *Data_type*: point, optional
`sdp_file` *`file_name`*	Specifies the name of the input file containing SDP definitions. The SDP file format supports wildcards (`*` and `?`) for instance names. It also supports numeric bus bit range as part of an instance name. *Data_type*: string, required
`-skip_placement`	Reads in the SDP group definition but does not place the group. *Data_type*: bool, optional

Examples

- The following command reads the `dtmf_chip.sdp` file and places all defined SDP elements. It also appends the specified hierarchical path to all elements.

```
read_sdp_file sdp_file dtmf_chip.sdp -hier_path DTMF_CHIP/TDSP_CORE_INST
```

Related Information

- `place_sdps`
- `write_sdp_file`
- `sdp` object in *Stylus Common UI Database Object Information*
- Using Structured Data Paths chapter of the *User Guide*
- Floorplan Menu chapter of the *Menu Reference*
    - *Structured Data Path*

set_sdp_origin

```
set_sdp_origin
[-help]
[-force]
-sdp group_name
{-location llx lly}
```

Specifies the location of a structured data path (SDP). If you know where the SDP will be placed in the design, use the `-location` option to specify the SDP location.

Before setting `set_sdp_origin -location`, you should have:

- Imported the design.
- Added members to the SDP group using the `update_sdp -add` option.

Parameters

`-help`	Prints a brief description that includes type and default information for each `set_sdp_origin` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man set_sdp_origin`.
`-force`	Indicates that the specified location should be set even if it is outside the existing core boundary. `-force` automatically adjusts the core boundary to accommodate the specified location. *Data_type*: bool, optional
`-location` *`llx lly`*	Specifies the location of the SDP top group. *Data_type*: point, optional
`-sdp` *`group_name`*	Sets the specified SDP group as the SDP top group. *Data_type*: string, required

Examples

- The following command specifies the location of the SDP group `tdsp_sdpGrp`:

```
set_sdp_origin -sdp tdsp_sdpGrp -location 848.82 586.75
```

Related Information

- `create_sdp`
- `update_sdp`
- `sdp` object in *Stylus Common UI Database Object Information*
- Using Structured Data Paths chapter of the *User Guide*
- Floorplan Menu chapter of the *Menu Reference*
  - *Structured Data Path*

update_sdp

```
update_sdp
[-help]
{[[[{-add {object_name_list} -sdp sdp_name} [-orient {r0 r90 r180 r270 mx mx90 my
my90}]] | -move {sdp(s) or inst(s)}] [-after refObject | -before refObject]] | -detach
{sdp(s) or inst(s)} | [-flip {sdp(s) or inst(s)} [-x  | -y ]]}
```

The `update_sdp` command enables you to perform update operations on a structured data path (SDP) object. You can use this command to:

- Add an object or member to an existing SDP group.
- Detach the specified SDP objects from their parent group.
- Flip an SDP group on X or Y axis along with its elements.
- Move SDP objects before or after the specified reference object.

In the traditional SDP flow, this command can be used while defining SDP constraints or while analyzing SDP alignment.

Parameters

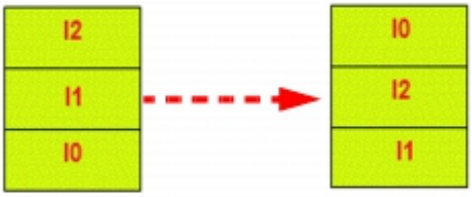
`-help`	Outputs a brief description that includes type and default information for each `update_sdp` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man update_sdp`.
`-add {object_name_list}`	Lists the names of sdps or instances to be added to the specified SDP group. The listed objects are added before or after the reference object, as specified with the `-before` or `-after` parameter, respectively. The `update_sdp -add` command enables you to create an SDP using TCL commands. You do not need to learn SDP relative placement format for creating a data path. Before adding a member to an SDP group, you should have imported the design and created an SDP group. For more information, see `create_sdp`. **Note**: The `-sdp` parameter must be specified along with this parameter. In addition, either `-before` or `-after` must be specified. *Data_type*: string, optional

`-after refObject`

Specifies a single sdp or instance as a reference object. The sdps or instances being added or moved are placed *after* the specified reference object.

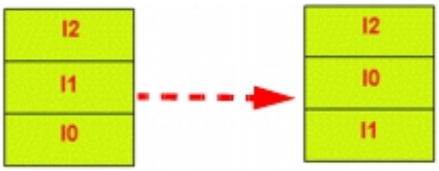
The interpretation of `-after` depends on the row or column's `justifyBy` direction. For example, consider the following command:
`update_sdp -move I0 -after I2`

- For a column with `justifyBy SW/SE`, `-after` means the specified SDP objects are placed above the reference object.



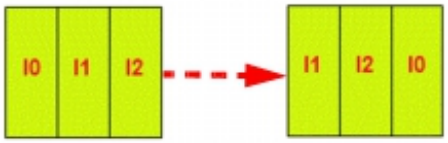
Column with `justifyBy SW/SE`

- For a column with `justifyBy NW/NE`, `-after` means the specified SDP objects are placed below the reference object.



Column with `justifyBy NW/NE`

- For a row with justifyBy SW/NW, `-after` means that the specified SDP objects are placed on the right side of the reference object.

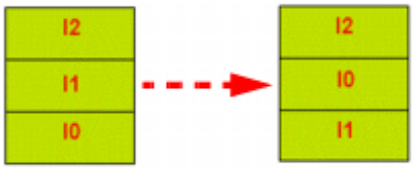
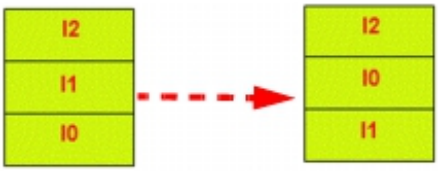
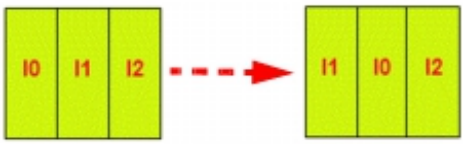
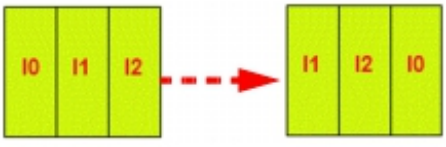


Row with `justifyBy SW/NW`

- For a row with justifyBy SE/NE, `-after` means that the specified SDP objects are placed on the left side of the reference object.

Row with `justifyBy SE/NE`

Data_type: string, optional

`-before` *refObject*	Specifies a single sdp or instance as a reference object. The sdps or instances being added or moved are placed *before* the specified reference object. The interpretation of `-before` depends on the row or column's `justifyBy` direction. For example, consider the following command: `update_sdp -move I1 -before I0` • For a column with `justifyBy SW/SE`, `before` means the specified SDP objects are placed below the reference object.  Column with `justifyBy SW/SE` • For a column with `justifyBy NW/NE`, `before` means the specified SDP objects are placed above the reference object.  Column with `justifyBy NW/NE` • For a row with justifyBy SW/NW, `before` means that the specified SDP objects are placed on the left side of the reference object.  Row with `justifyBy SW/NW` • For a row with justifyBy SE/NE, `before` means that the specified SDP objects are placed on the right side of the reference object.  • Row with `justifyBy SE/NE` *Data_type*: string, optional

`-detach {sdp(s) or inst(s)}`	
	Detaches the specified SDP object(s) from their parent group. *Data_type*: string, optional
`-flip {sdp(s) or inst(s)}`	
	Flips the specified SDP groups on the X or Y axis, along with their elements. The alignment information (`justifyBy`) is adjusted accordingly. Flipping can be used to reduce wire length. **Note**: Use the `-x` or `-y` parameter to specify the axis for flipping. The default axis for flipping is Y . *Data_type*: string, optional
`-move {sdp(s) or inst(s)}`	
	Lists the names of SDP objects to be moved. The listed objects are moved before or after the reference object, as specified with the `-before` or `-after` parameter, respectively. You can use the `-move` parameter to: • Move a subgroup or member to a different position within the current SDP group. • Move a subgroup or member to a different group to improve congestion and/or timing. • Move multiple instances, SDP groups, or a mix of instances and SDP groups to the specified location. The `-move` parameter supports the movement of multiple instances, SDP groups, or a mix of these at the same time. If multiple SDP objects are specified with `-move`, the option works as follows: • If both the objects being moved and the reference object are instances, the instances being moved will become members of the SDP group to which the reference instance belongs. • If both the objects being moved and the reference object are SDP groups, the SDP groups being moved will become members of the SDP group to which the reference SDP group belongs.

	• If the objects being moved are SDP groups and the reference object is an instance, the SDP groups being moved will become members of the SDP group to which the reference instance belongs. • If the objects being moved are instances and the reference object is an SDP group, the instances being moved will become members of the SDP group to which the reference SDP group belongs. • If the objects being moved are a mix of instances and SDP groups and the reference object is an instance, the instances and SDP groups being moved will become members of the SDP group to which the reference instance belongs and the result is based on the input sequence. • If the objects being moved are a mix of instances and SDP groups and the reference object is an SDP group, the instances and SDP groups being moved will become members of the SDP group to which the reference SDP group belongs and the result is based on the input sequence. **Note**: Either `-before` or `-after` must be specified with the `-move` parameter. *Data_type*: string, optional
`-orient {r0 r90 r180 r270 mx mx90 my my90}`	
	Specifies the orientation to be set for the object being added or moved. *Default*: `r0` (no rotation) *Data_type*: enum, optional
`-sdp` *sdp_name*	Specifies the name of the SDP group to which you are adding the member. **Note**: The `-add` parameter must be specified along with this parameter. *Data_type*: string, optional
`-x`	Flips the SDP group along the X axis. **Note**: The `-flip` parameter must be specified along with this parameter. *Data_type*: bool, optional

`-y`	Flips the SDP group along the Y axis. **Note**: The `-flip` parameter must be specified along with this parameter. *Data_type*: bool, optional

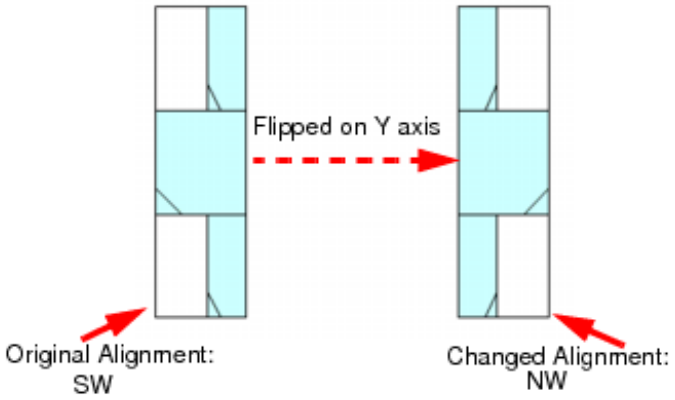
Examples

- The following examples adds two objects, `DTMF_INST/tdsp_core_inst/inst1` and `DTMF_INST/tdsp_core_inst/inst2`, to the `sdpGrp1` group.

```
update_sdp -add {DTMF_INST/tdsp_core_inst/inst1 DTMF_INST/tdsp_core_inst/inst2} -
sdp sdpGrp1 -after DTMF_INST/PLLCLK_INST
```

- The following command flips the SDP group `sdpGrp1` on the Y axis.

```
update_sdp -flip sdpGrp1 -y
```



- The following command moves the SDP group `sdpGrp1`, which is an SDP group of type `row` with `justifyBy SE`, to the right side of the group `sdpGrp3`:

```
update_sdp -move sdpGrp1 -before sdpGrp3
```

- The following command moves the SDP group `SDP_col145`, which is an SDP group of `type column` with `justifyBy NE`, to the position below `SDP_col170`.

```
update_sdp -move SDP_col145 -after SDP_col170
```

- Consider the SDP group structure given below:

```
column group1 {
   justifyBy SW
   row selInstGrp_9 {
      justifyBy SW
```

```
      inst DTMF_INST/TDSP_CORE_INST/EXECUTE_INST/ar1_reg_1
      inst DTMF_INST/TDSP_CORE_INST/EXECUTE_INST/ar1_reg_13 orient MX
      inst DTMF_INST/TDSP_CORE_INST/EXECUTE_INST/ar1_reg_12
      inst DTMF_INST/TDSP_CORE_INST/EXECUTE_INST/ar1_reg_3
      inst DTMF_INST/TDSP_CORE_INST/EXECUTE_INST/ar0_reg_12 orient R0
      inst DTMF_INST/TDSP_CORE_INST/EXECUTE_INST/ar0_reg_13 orient MX
      inst DTMF_INST/TDSP_CORE_INST/PORT_BUS_MACH_INST/data_out_reg_8
      inst DTMF_INST/TDSP_CORE_INST/EXECUTE_INST/ar1_reg_14
   }
   .....
}
```

- The following command moves the specified instances before `selInstGrp_9`:

  ```
  update_sdp -move {DTMF_INST/TDSP_CORE_INST/EXECUTE_INST/acc_reg_30
  DTMF_INST/TDSP_CORE_INST/EXECUTE_INST/acc_reg_32} -before selInstGrp_9
  ```

  The resulting structure is as follows:

  ```
  column group1 {
     justifyBy SW
     inst DTMF_INST/TDSP_CORE_INST/EXECUTE_INST/acc_reg_30
     inst DTME_INST/TDSP_CORE_INST/EXECUTE_INST/acc_reg_32
     row selInstGrp_9 {
        justifyBy SW
        inst DTMF_INST/TDSP_CORE_INST/EXECUTE_INST/ar1_reg_1
        inst DTMF_INST/TDSP_CORE_INST/EXECUTE_INST/ar1_reg_13 orient MX
        inst DTMF_INST/TDSP_CORE_INST/EXECUTE_INST/ar1_reg_12
        inst DTMF_INST/TDSP_CORE_INST/EXECUTE_INST/ar1_reg_3
        inst DTMF_INST/TDSP_CORE_INST/EXECUTE_INST/ar0_reg_12 orient R0
        inst DTMF_INST/TDSP_CORE_INST/EXECUTE_INST/ar0_reg_13 orient MX
        inst DTMF_INST/TDSP_CORE_INST/PORT_BUS_MACH_INST/data_out_reg_8
        inst DTMF_INST/TDSP_CORE_INST/EXECUTE_INST/ar1_reg_14
     }
     .....
  }
  ```

- The following command moves the specified instances after `selInstGrp_9`:

  ```
  update_sdp -move {DTMF_INST/TDSP_CORE_INST/EXECUTE_INST/acc_reg_30
  DTMF_INST/TDSP_CORE_INST/EXECUTE_INST/acc_reg_32} -after selInstGrp_9
  ```

  The resulting structure is as follows:

  ```
  column group1 {
     justifyBy SW
     row selInstGrp_9 {
        justifyBy SW
        inst DTMF_INST/TDSP_CORE_INST/EXECUTE_INST/ar1_reg_1
  ```

```
        inst DTMF_INST/TDSP_CORE_INST/EXECUTE_INST/ar1_reg_13 orient MX
        inst DTMF_INST/TDSP_CORE_INST/EXECUTE_INST/ar1_reg_12
        inst DTMF_INST/TDSP_CORE_INST/EXECUTE_INST/ar1_reg_3
        inst DTMF_INST/TDSP_CORE_INST/EXECUTE_INST/ar0_reg_12 orient R0
        inst DTMF_INST/TDSP_CORE_INST/EXECUTE_INST/ar0_reg_13 orient MX
        inst DTMF_INST/TDSP_CORE_INST/PORT_BUS_MACH_INST/data_out_reg_8
        inst DTMF_INST/TDSP_CORE_INST/EXECUTE_INST/ar1_reg_14
    }
    inst DTMF_INST/TDSP_CORE_INST/EXECUTE_INST/acc_reg_30
    inst DTME_INST/TDSP_CORE_INST/EXECUTE_INST/acc_reg_32
    .....
}
```

Related Information

- add_sdp_rows_blockages
- create_sdp
- delete_sdp_rows_blockages
- sdp object in *Stylus Common UI Database Object Information*
- Using Structured Data Paths chapter of the *User Guide*
- Floorplan Menu chapter of the *Menu Reference*
    - *Structured Data Path*

write_sdp_file

```
write_sdp_file
[-help]
out_file
[-template | -top_group group_name]
[-template | -command]
[-template  | [-update_relative_place_constraint | -update_orientation]]
```

Saves the current SDP relative placement information into a file. By default, the output is in `.sdp` format. `write_sdp_file` enables you to save SDP relative information so it can be reloaded in another Innovus session.

The `write_sdp_file` command can be run any time after SDPs have been defined.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `write_sdp_file` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man write_sdp_file`.
`-command`	Specifies that the relative placement information should be saved in TCL commands format instead of `.sdp` format. *Data_type*: bool, optional
out_file	Specifies the name of the output file containing SDP relative information. *Data_type*: string, required
`-template`	Specifies that the file should be saved as a template in `.sdp` format. **Note:** This parameter cannot be used with `-command` or `-top_group`. *Data_type*: bool, optional

`-top_group` *group_name*	Indicates that the relative placement information of only the specified SDP group should be written. *Data_type*: string, optional
`-update_orientation`	Updates the orientation of SDP instances before writing them to the SDP file. *Data_type*: bool, optional
`-update_relative_place_constraint`	Updates relative placement constraints of SDPs before writing out the SDP file. Use this parameter to write out the updated SDP/TCL file after any change in SDP relative placement due to `placeDesign` or manual changes. The `-update_relative_place_constraint` option can be used with the `-command` and `-topGroup` options as well. *Data_type*: bool, optional

Examples

- The following command writes the current SDP information in SDP format in a file named `test.sdp`:

  ```
  write_sdp_file test
  ```

- The following command writes the SDP information for the `sdpGrpA` group in the SDP TCL command format to the output file `output.sdp`.

  ```
  write_sdp_file output.sdp -command -top_group sdpGrpA
  ```

Related Information

- `place_sdps`
- `read_sdp_file`
- `sdp` object in *Stylus Common UI Database Object Information*
- Using Structured Data Paths chapter of the *User Guide*
- Floorplan Menu chapter of the *Menu Reference*

- *Structured Data Path*

36

Integrated Hierarchical Database Commands

- check_module_model
- define_add_on_module_model
- import_module_model
- pack_module_model_libs
- read_module_model
- report_module_model
- set_module_model
- update_module_model
- write_module_model
- write_module_model_context

check_module_model

```
check_module_model
[-help]
-cell cell_name
[-out_file file_name]
-ref_tag tag_name
[-slack_threshold threshold] [-type {flexIlm ilm pnr}]
```

Checks and validates an ILM model to see if the current reduction ratio will cause any missing constraints at the hinst terminal or any slack difference. The `check_module_model` command compares/checks the target timing result with the reference timing result and generates a summary and detailed report.

Note: If a specified cell has more than one instantiated modules, all hinst terminals are checked and reported.

Usage Model:

- Generate a reference ILM model with the `write_ilm -keep_all` option and a target ILM model with user-specified settings.
- Restore the top-level design and set/commit target ILM model(s).
- Time the design.
- Call `check_module_model`

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `check_module_model` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man check_module_model`
`-cell` *`cell_name`*	Specifies the ILM cell name that will be validated. *Data_type*: string, required
`-out_file` *`file_name`*	Specifies the output report file. *Default:* `./check_module_model.rpt` *Data_type*: string, optional
`-ref_tag` *`tag_name`*	Specifies the reference tag name of the reference ILM model. *Data_type*: string, required
`-slack_threshold` *`threshold`*	Specified the slack threshold for reporting the number of mismatched slacks over the threshold. *Default:* `0.0005` *Data_type*: string, optional
`-type {flexIlm ilm pnr}`	Specifies the type of module model. The model type can be flexIlm, ilm, or pnr. *Default*: `ilm` *Data_type*: string, optional

Example

- The following command check the tdsp_core cell and writes the detailed report in the tdsp_check_module_model.rpt file:

```
check_module_model -cell tdsp_core -ref_tag ref_preCts -type ilm  -out_file
tdsp_check_module_model.rpt
```

Related Information

- `commit_module_model`
- `write_module_model`
- `read_module_model`
- `write_partitions`
- `set_module_model`
- `update_module_model`
- Hierarchical and Prototyping Flow chapter in the *Innovus Stylus Common UI User Guide*
  - Integrated Hierarchical Database Flow

define_add_on_module_model

```
define_add_on_module_model
[-help]
-type string
[-commit_callback string]
{[-create_callback string]
[-set_callback string]
[-read_callback string]
[-write_context_callback string]}
```

Allows you to define the add_on types or supplemental models that can be loaded additionally with an existing model or a design. These user-defined add_on types can be used with the `write_module_model`, `set_module_model`, `read_module_model`, and `write_module_model_context` commands. You must provide a TCL file containing callback procedures for each applicable iHDB command that will be run when this command is called.

In the TCL file, the TCL callback procedures must be defined in the following format and must include the specified arguments:

```
proc <callBackTCLProcName> {dir cell tag args} {
…
}
```

Where,

`dir`, `cell`, and `tag` represent values that will be automatically derived by the iHDB capability.
`args` can be set using `set_module_model -default_commit_options {options} -type add_ons_type`

Parameters

`-commit_callback` *string*	Specifies the user-defined TCL procedure that is executed when the `commit_module_model` command is called with this `add_ons` type. *Data_type*: string, optional
`-create_callback` *string*	Specifies the user-defined TCL procedure that is executed when the `write_module_model` command is called with this `add_ons` type. *Data_type*: string, optional
`-help`	Outputs a brief description that includes the type and default information for each `define_add_on_module_model` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man define_add_on_module_model`
`-read_callback` *string*	Specifies the user-defined TCL procedure that is executed when the `read_module_model` command is called with this `add_ons` type. *Data_type*: string, optional
`-set_callback` *string*	Specifies the user-defined TCL procedure that is executed when the `set_module_model` command is called with this `add_ons` type. *Data_type*: string, optional
`-type` *string*	Specifies the newly added add_ons model type name. *Data_type*: string, required
`-write_context_callback` *string*	Specifies the user-defined TCL procedure that is executed when the `write_module_model_context` command is called with this `add_ons` type. *Data_type*: string, optional

Example

- Following is an example of defining the "attrs" add_ons model type. First this 'attrs' attribute must be defined with the `define_add_on_module_model` command with its associated callback TCL procedures:

  ```
  source dump_attrs.tcl
  set_module_model -default_commit_options {-encounter_format -keep_pin_geometry} -type pnr
  define_add_on_module_model -type attrs -create_callback ihdb_create_attrs -set_callback
  ihdb_read_attrs
  ```

  Where `dump_attrs.tcl` contains:

  ```
  proc ihdb_create_attrs {dir cell tag args} {   ;#This format is fixed

    write_block_attrs $dir/$cell/$tag/attrs/db.attrs

  }

  proc ihdb_read_attrs {dir cell tag args} {  ;# This format is fixed

    read_block_attrs $dir/$cell/$tag/attrs/db.attrs

  }

  …
  ```

- *Block Level*. The following command creates user defined attrs type of module model that has the *merge_attribute* tag.

  ```
  write_module_model -type attrs -tag merge_attribute

  write_module_model -tag merge_attribute
  ```

- *Top-Level*. The following commands configures and commits the *merge_attribute* module model using the `attrs` add-on associated with the `pnr` model type:

  ```
  set_module_model -cell dcm -tag merge_attribute -type pnr -add_ons {attrs}
  set_module_model -cell lsu -tag merge_attribute -type pnr -add_ons {attrs}
  commit_module_model
  ```

Related Information

- `commit_module_model`
- `write_module_model`
- `read_module_model`
- `write_partitions`
- `set_module_model`
- `update_module_model`
- Hierarchical and Prototyping Flow chapter in the *Innovus Stylus Common UI User Guide*
  - Integrated Hierarchical Database Flow

import_module_model

```
import_module_model
[-help]
-cell string
[-idx_pair string]
-inputs string
-libs [-link_only]
[-rc_corner string]
[-tag string]
-type {lib lef qrc_tech qrc_conf}
```

Imports library files into the iHDB structure.

Parameters

`-cell` *string*	Specifies the cell to import. *Data_type*: string, required
`-help`	Outputs a brief description that includes the type and default information for each `import_module_model` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man import_module_model`
`-idx_pair` *string*	Specifies the idx pair list if the files are copied from the idx database. *Data_type*: string, optional
`-inputs` *string*	Specifies the full path of the input file to be imported. *Data_type*: string, required
`-libs`	Specifies whether input file is a library file or not. **Note:** This option should be used for type lef, lib, qrc_tech, or qrc_conf. **Note:** The library files: • *.lib are saved at <iHDB>/<cell>/.libs/mmmc/ • *.lef are saved at <iHDB>/<cell>/.libs/lef/ • qrcTechFile are saved at <iHDB>/<cell>/.libs/mmmc/<rc_corner>/qrcTechFile • qrc.conf are saved at <iHDB>/<cell>/.libs/qx/qrc.conf *Data_type*: bool, required

`-link_only`	Creates a link or actual hard copy. **Note:** If the `-link_only` parameter is specified, then a link pointing to the current library location is created. If is the `-link_only` parameter is not specified then the actual physical file will be there. *Data_type*: bool, optional
`-rc_corner` *string*	Specifies the list of RC corners. *Data_type*: string, optional
`-tag` *string*	Specifies a tag of a saved file. **Note:** The `-tag` and `-libs` parameters are mutually exclusive. *Data_type*: string, optional
`-type {lib lef qrc_tech qrc_conf}`	
	Specifies the type(s) of module model. The model type can be lib, lef, qrc_tech, or qrc_conf. Default: *Data_type*: string, required

Example

- The following command imports a LEF file:
```
import_module_model -cell tdsp_core -libs -type lef -input_file
/icd/hierflow2/DATA2/tdsp_core/init/pnr/tdsp_core.enc.dat/libs/lef/*.lef
```

Related Information

- `set_module_model`
- Hierarchical and Prototyping Flow chapter in the *Innovus Stylus Common UI User Guide*
  - Integrated Hierarchical Database Flow

pack_module_model_libs

```
pack_module_model_libs
[-help]
[-cell cellName]
[-validate]
```

Packs libraries for Integrated Hierarchical Database (iHDB) designs.

Notes:

- You must specify the default central directory name (`set_module_model -default_dir`), before using this command. In case default central directory is moved to a different location, the libraries can be packed again.
- The packed libraries are saved at *<default_model_data_directory>/<cell>/.libs.*

Parameters

`-cell` *cellName*	Specifies the cell name. All pnr model libraries of all tags under the specified cell will be linked to the packed libraries (*.libs)* saved by the `pack_module_model_libs` command. **Note:** The packed libraries are saved at *<default_model_data_directory>/<cell>/.libs.* *Data_type*: string, optional
`-help`	Outputs a brief description that includes the type and default information for each `pack_module_model_libs` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man pack_module_model_libs`
`-validate`	Compares the checksum before and after packing libraries. It checks whether the libraries before and after packing are the same. **Note:** When the `-validate` parameter is specified, the `pack_module_model_libs` command will output *after_pack_libs.check_sum* and *before_pack_libs.check_sum* under the *.libs* directory for comparison. *Data_type*: bool, optional

Example

- The following command specifies the default central directory name (./DATA) where models data will be saved/loaded and then specifies the cell name. All pnr model libraries of all tags under the specified cell will be linked to the packed libraries (*.libs)* saved by the `pack_module_model_libs` command.

  ```
  set_module_model -default_dir DATA2
  pack_module_model_libs -cell tdsp_core -validate
  Output:
  ```

  This command will output the packed libraries at *`DATA2/tdsp_core/.libs`*. It will also create *`after_pack_libs.check_sum`* and *`before_pack_libs.check_sum`* under the *`.libs`* directory for comparison.

Related Information

- set_module_model
- Hierarchical and Prototyping Flow chapter in the *Innovus Stylus Common UI User Guide*
  - Integrated Hierarchical Database Flow

read_module_model

```
read_module_model
[-help]
[-dynamic_view string]
[-hold_views string]
[-leakage_view string]
[-lef_files string]
[-mmmc_file string]
[-no_timing]
[-no_timing_graph]
[-setup_views string]
[-tag tag_name]
{cell_name | -cell cell_name}
[options]
[-add_ons {1801 mmmc spef eco latency scan_info timing_context extraction_context pin_assign}]
```

Restores a design from a default module model repository. You can use this command only after the default central directory name where the module model data will be saved/loaded has been specified using the `set_module_model -default_dir` command.

The `read_module_model` command provides you the following flexibilities:

- Ability to provide your own ECO script. The ECO script must be located at <cell>/<tag>/eco/<cell>_eco.tcl.
- Ability to read/restore a design of a specific tag but its ECO file can be loaded from a different tag.

This feature is useful when the ECO changes are done with an old netlist drop and a new netlist drop is saved with a different tag.

For example:

```
read_module_model -cell tdsp_core -tag my_postcts -add_ons {anotherTag/eco}
```

Supported short-hand `read_module_model` usage model:

```
read_module_model <dir>/<cell>/<tag> [-add_ons {add_on_type}]
read_module_model <cell>/<tag> [-add_ons {add_on_type}] ;# Local directory
read_module_model <cell> -tag <tag> [-add_ons {add_on_type}] ;# Default root dir
```

Parameters

```
-add_ons {1801 mmmc spef eco latency scan_info timing_context extraction_context pin_assign}
```

	Specifies the supplement model that can be loaded additionally with an existing model or a design. For example: • The following command restores the ptn_wrapper design that has my_extract_v1 tag and also bring in its extraction_context add_on model: `read_module_model ptn_wrapper -tag my_extract_v1 -add_ons {extraction_context}` • The following commands show the location from where mmmc data will be loaded during `read_module_model` according to usage of `-add_ons {mmmc}` ◦ `read_module_model Ptn1 -tag XYZ` Uses view definition located inside .enc.dat in pnr directory of tag XYZ ◦ `read_module_model Ptn1 -tag XYZ -add_ons {ABCD/mmmc}` Uses view definition from mmmc directory of tag ABCD ◦ `read_module_model Ptn1 -tag XYZ -add_ons {mmmc}` Uses view definition from mmmc directory of tag XYZ **Note:** For more information about saving data in separate tags in the mmmc directory, see write_timing_budgets `-module_model_tag` *Data_type*: string, optional
cell_name | `-cell` *cell_name*	Specifies the module model name. **Note:** This parameter must be specified. *Data_type*: string, required
`-dynamic_view` *string*	Specifies the dynamic view. *Data_type*: string, optional
`-help`	Outputs a brief description that includes the type and default information for each `read_module_model` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man read_module_model`
`-hold_views` *string*	Specifies the hold views. **Note:** The `-setup_views` and `-hold_views` parameters specify the active analysis views for the loaded design. *Data_type*: string, optional
`-leakage_view` *string*	Specifies the leakage view. *Data_type*: string, optional

`-lef_files` *string*	Specifies the lef file(s). *Data_type*: string, optional
`-mmmc_file` *string*	Specifies the MMMC view file(s) *Data_type*: string, optional
`-no_timing`	Ignores timing libraries and MMMC configuration *Data_type*: bool, optional
`-no_timing_graph`	Specifies not to load timing graph. *Data_type*: bool, optional
options	Specifies the read_db options. *Data_type*: string, optional
`-setup_views` *string*	Enables you to setup views. **Note:** The `-setup_views` and `-hold_views` parameters specify the active analysis views for the loaded design. *Data_type*: string, optional
`-tag` *tag_name*	Specifies the tag name of the flow. **Note:** If *tag_name* is not specified, 'latest' is used as the default tag value. *Data_type*: string, optional

Example

- The following command restores the dtmf_recvr_core module model that has the start tag.
  ```
  read_module_model -cell dtmf_recvr_core -tag start
  ```
- The following commands restore the ptn_wrapper design along with its DEF extraction context to do hierarchical extraction:
  ```
  read_module_model ptn_wrapper -tag my_extract_v1 -add_ons {extraction_context}
  #Load in the detailed LEF abstract of ram_128x16_test
  set_module_model -cell ram_128x16_test -type lef -tag my_extract_v1
  commit_module_model
  route_early_global
  set_db extract_rc_engine post_route
  extract_rc
  write_parasitics -spef_file with_detailed_def.spef
  ```

Related Information

- commit_module_model
- write_module_model
- write_partitions
- set_module_model
- update_module_model
- Hierarchical and Prototyping Flow chapter in the *Innovus Stylus Common UI User Guide*
  - Integrated Hierarchical Database Flow

report_module_model

```
report_module_model
[-help]
[-active_rc_corners]
[-default_dir]
[-non_committed_hinst]
[-type {flexilm lef ilm pnr pnr_shell}]
```

Reports the module models that are currently loaded/committed at top-level design.

Parameters

`-active_rc_corners`	Reports the active RC corners. *Data_type*: bool, optional
`-default_dir`	Reports the current default module data directory. *Data_type*: bool, optional
`-help`	Outputs a brief description that includes the type and default information for each `report_module_model` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man report_module_model`
`-non_committed_hinst`	Checks and reports the modules/cells that are available in the default module data directory but are not committed as hierarchical modules (hInst). *Data_type*: bool, optional
`-type {flexilm lef ilm pnr pnr_shell}`	Specifies the type of module model to be reported **Note:** The module models that are not committed are reported with the "init" type and tag. The module models that are committed are reported with the types and tags same as the requested types. *Data_type*: string, optional

Examples

- The following command reports the module models that are currently loaded/committed at top-level design:

```
report_module_model -type lef

Module       Type ReqType Tag     ReqTag    Dir
-------------------------------------------------------------------------------------
ptn_wrapper  lef  lef     preCts  preCts    /icd/demo/v4h/DATA2/ptn_wrapper/latest/lef


default directory : /icd/demo/v4h/DATA2
```

- The following command reports the ilm type of module models:

```
report_module_model -type ilm

Module       Type ReqType Tag         ReqTag      Dir
-------------------------------------------------------------------------------------------
ptn_wrapper  ilm  ilm     my_postCts  my_postCts  /icd/demo/v4h/DATA2/ptn_wrapper/my_postCts/ilm
tdsp_core    ilm  ilm     my_preCts   my_preCts   /icd/demo/v4h/DATA2/tdsp_core/my_preCts/ilm

default directory : /icd/demo/v4h/DATA2
```

- The following command reports the list of active RC corners:

```
set_module_model -active_rc_corners {rcMin rcMax}
report_module_model -active_rc_corners
```

Related Information

- commit_module_model
- write_module_model
- write_partitions
- set_module_model
- update_module_model
- Hierarchical and Prototyping Flow chapter in the *Innovus Stylus Common UI User Guide*
  - Integrated Hierarchical Database Flow

set_module_model

```
set_module_model
[-help]
[-allow_port_mismatch]
[-dir model_data_directory]
[-hold_views string]
[-phys_hier]
[-setup_views string]
[-tag flow_tag]
[-type {flexilm lef ilm pnr pnr_shell}]
[-update]
{[[[-cell moduleName] [-default_set_options string]] | [-top cell_name [-logical_only ]]]
[-add_ons {1801 boundary_model etm latency pnr_view scan_info spef timing_context
extraction_context}]] |
-default_dir default_model_data_directory |
-default_create_options default_option |
-default_commit_options string |
-active_rc_corners list_of_rc_corners }
```

Sets or configures specific block model(s) to top-level design.

Parameters

`-active_rc_corners list_of_rc_corners`	Specifies the list of RC corners. **Note:** For multi-corners, different applications have different ways to get or specify RC corners for outputting/reading SPEF (Standard Parasitic Exchange Format). In the iHDB flow, the `-active_rc_corners` parameter provides a uniform way to use RC corners. **Note:** The `write_module_model -type spef` honors the `set_module_model -active_rc_corners` command to output SPEF(s) for the specified RC corners. By default, multi corner SPEFs are saved as gzip. The output SPEF(s) are saved at <SHDB_root>/<cell>/<tag>/spef/<topCell>. <cornerName>.spef.gz *Data_type*: string, optional
`-add_ons {1801 boundary_model etm latency pnr_view scan_info spef timing_context extraction_context}`	

	Specifies the supplement model that can be loaded additionally with an existing model or a design. The following is the list of the supported model types: • `1801`: UPF for power domain • `boundary_model`: Boundary model for timing analysis • `etm`: Extraction timing model • `latency`: Clock latency information • `pnr_view`: Full pnr physical DB • `scan_info`: Scan chain information • `spef`: SPEF file from RC extraction • `timing_context`: Timing context model • `extraction_context`: Extraction context model For example the following command specifies that the etm add on type is associated with the `-type lef` parameter. It uses LEF and timing model ETM (.lib) together at top-level design: `set_module_model -cell tdsp_core -type lef -add_ons {etm}` **Note:** The `1801` add on type can only be specified for `ilm` model type (`-type ilm`). **Note:** The `pnr_view` add on type can only be specified for `ilm` model type (`-type ilm`) where it will load in the full pnr physical DB on top of an ILM model so you can see all cells and routing. This can be used for debugging timing. **Note:** The `spef` add on type honors the `-active_rc_corners` parameter to read in SPEFs (Standard Parasitic Exchange Format) for the specified RC corners. **Note:** The `spef` add on type cannot be used with the `pnr_shell` model type. **Note:** With the `latency` and `scan_info` add on types, the `write_module_model` command can save the latency and scan chain data into iHDB. The `commit_module_model` command can bring back the latency data of an ILM data to the top-level design for pre-CTS stage, and be able to merge block scan chains back to top-level design. *Data_type*: bool, optional

`-allow_port_mismatch`	Allows model I/O ports mismatching from the original database. It replaces the new updated LEF and provides an ability to bring in an updated model with mis-matched ports to top-level design. **Note:** This parameter can be specified with the lef, flexllm, ilm, and pnr type (`-type`) of module models only. *Data_type*: string, optional
`-cell` *moduleName*	Specifies the cell name(s) to be loaded. You can specify a list of cell names. **Note:** The `-cell` parameter accepts wildcards as input. *Data_type*: string, optional

`-default_commit_options` *string*	Specifies the default commit options. These default settings are honored by the `commit_module_model` command.
	Use Model:
	`set_module_model -default_commit_options {listOf_assemble_design_options} -type pnr`
	`set_module_model -default_commit_options {listOf_set_ilm_type_Options} -type ilm`
	Example:
	`set_module_model -default_commit_options {-encounter_format -keep_pin_geometry} -type pnr`
	`set_module_model -default_commit_options {-model si} -type ilm`
	Data_type: string, optional

`-default_create_options` *default_options*	Specifies the default create options for a specific model type. These default settings are honored by the `write_module_model` command. **Note:** You must specify the `-type` parameter with this option. **Note:** While specifying the default create options, ensure that you specify the correct names of the options as per the selected Innovus interface. *Data_type*: string, optional
`-default_set_options` *string*	Sets the default options. *Data_type*: string, optional

`-default_dir` *default_model_data_directory*	Specifies the default central directory name where the module models data will be saved/loaded. **Note:** If `set_module_model -default_dir` is not specified, the Innovus system does not set any directory by default. Later on, when you try to run the `read_module_model` command, it displays an error message that the `-default_dir` parameter is not set. This option MUST be set before loading any Integrated Hierarchical Database (iHDB) design. *Data_type*: string, optional
`-dir` *model_data_directory*	Specifies the directory where data of the specified cell can be found. **Note:** The `-dir` parameter will override the `-default_dir` setting for this cell. **Note:** The `-dir` parameter cannot be specified if the `-cell` parameter has been specified using wildcards. *Data_type*: string, optional
`-help`	Outputs a brief description that includes the type and default information for each `set_module_model` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man set_module_model`
`-hold_views` *string*	Specifies the hold views. **Note:** The `-setup_views` and `-hold_views` parameters allow you to specify the active analysis views for the loaded design. Innovus uses these settings to know which SDC constraints to use. By default, the active views will be based on the viewDefinition.file of pnr model (*.enc.dat). *Data_type*: string, optional
`-logical_only`	Specifies that only the logical data will be read. It controls whether to load DEF and LEF libraries or not. If `-logical_only` is specified, Innovus only loads Verilog/.lib but not DEF/LEF. *Data_type*: bool, optional
`-phys_hier`	Enables you to load the physical hierarchy for viewing purpose. *Data_type*: bool, optional

`-setup_views` *string*	Enables you to setup views. **Note:** The `-setup_views` and `-hold_views` parameters allow you to specify the active analysis views for the loaded design. Innovus uses these settings to know which SDC constraints to use. By default, the active views will be based on the viewDefinition.file of pnr model (*.enc.dat). *Data_type*: string, optional
`-tag` *flow_tag*	Specifies the tag name of the specified cell. It can be design stage/status name such as ‘prects’, ‘cts’, etc. **Note:** The 'latest' tag will be used if this option is not specified. ‘latest’ can also be specified as tag name for referencing to the latest updated tag. **Note:** For a specific cell, you only need to specify this tag option for one model type. For subsequent model type setting of the same cell, same tag name will be used if it’s not specified. *Data_type*: string, optional
`-top` *cell_name*	Specifies the top module model. When specified it configures the Verilog netlist and/or the DEF file of the specified top-level design. **Note:** When the `-top` parameter is specifies, the default model type (`-type`) is pnr.
`-type {flexilm lef ilm pnr pnr_shell}`	Specifies the model type. The model type can be lef, flexllm, ilm, pnr, or pnr_shell. • `flexilm`: Flexible interface logic model • `ilm`: Interface logic model • `lef`: LEF abstract • `pnr`: Innovus DB • `pnrshell`: Reduced pnr model where only the first level of logics is kept. **Note:** You must specify the `-type` parameter. *Data_type*: enum, optional
`-update`	Specifies that the data of the specified cell will be loaded without doing any checking. *Data_type*: bool, optional

Example

- The following command specifies the default central directory name (./DATA) where models data will be saved/loaded.

  ```
  set_module_model -default_dir ./DATA
  ```

- The following command configures the tdsp_core module model using LEF and timing model ETM (.lib) together at the top-level design:

  ```
  set_module_model -cell tdsp_core -tag my_prects -type lef -add_ons {etm}
  ```

- The following example specifies the default options for type lef:

  ```
  set_module_model -type lef -default_create_options {-top_layer M10 -stripe_pins}
  ```

Related Information

- `write_module_model`
- `commit_module_model`
- `read_module_model`
- `write_partitions`
- `update_module_model`
- Hierarchical and Prototyping Flow chapter in the *Innovus Stylus Common UI User Guide*
  - Integrated Hierarchical Database Flow

update_module_model

```
update_module_model
[-help]
[-num_cpu number_of_cpus]
[-distribute_job]
[-eco_report]
[-log_dir dir_name]
[-return_tag tag_name]
[-return_type {lef flexIlm ilm pnr}]
[-tag tag_name]
{[-distribute_run | -cell cellName] [-interactive | -plugin_tcl tcl_commands | -generate_eco_only ]}
```

Invokes an independent session in interactive or batch mode at top-level design. The `update_module_model` command can be called at top-level design.

Note: When an independent session is invoked, the license(s) will be checked-out from the main Innovus session and checked in to the independent session. Once you exit the independent session, the license(s) will return back to main Innovus session.

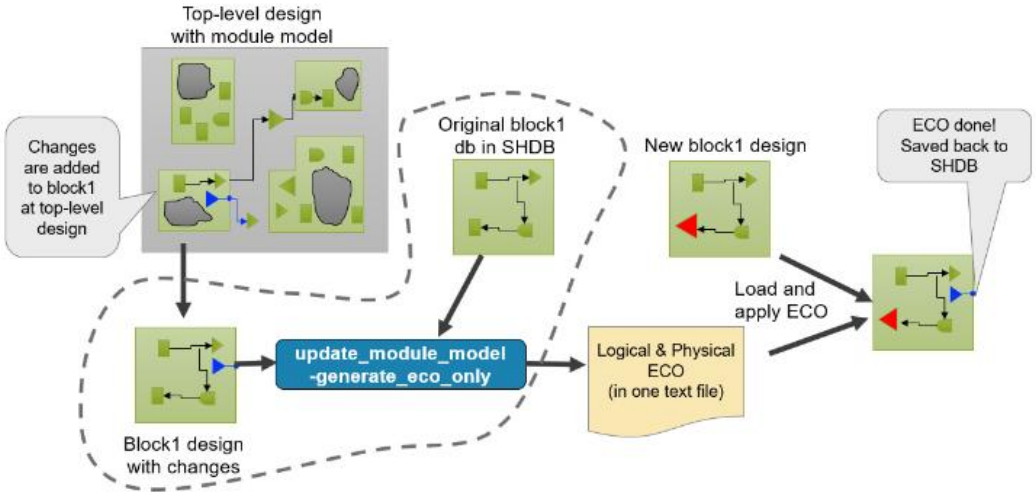
The `update_module_model` command:

- automatically updates the changes from top to block in an independent session.
- runs the user-provided plug-in commands in batch mode.
- automatically brings back the updated LEF models of the same tag to the top-level once the job is done.

Note: You should run the `write_module_model` command to save different types of models at the end of an independent session.

Parameters

`-cell` *cell_name*	Specifies which cell the command will update changes. *Data_type*: string, optional
`-distribute_job`	Enables distributed processing where more than one block design can be run in parallel. *Data_type*: bool, optional
`-distribute_run`	Enables you to execute all the specified `update_module_model` distributed jobs using a Makefile. *Data_type*: bool, optional
`-eco_report`	Specifies the output report file that contains the ECO change information. *Data_type*: bool, optional

`-generate_eco_only`	Generates an eco file only that contains general netlist ECO and placement changes. This mode allows you to explicitly review the ECO before applying it to the block DB.  *Data_type*: bool, optional
`-help`	Outputs a brief description that includes the type and default information for each `update_module_model` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man update_module_model`
`-interactive`	Invokes independent session in interactive mode. *Data_type*: bool, optional
`-log_dir` *dir_name*	Specifies the log file directory name. *Data_type*: string, optional
`-num_cpu` *number_of_cpus*	Specifies the number of CPUs that will be used for the independent session. By default it uses the same of number of CPUs as its parent. *Data_type*: int, optional
`-plugin_tcl` *tcl_commands*	Specifies the TCL commands to be run in a batch independent session. *Data_type*: string, optional
`-return_tag` *tag_name*	Specifies the tag name that will be used to update the specified cell model at the top-level design. *Default*: The default return tag is the tag name of the current loaded block design. *Data_type*: string, optional
`-return_type {flexIlm ilm lef pnr}`	

	Specifies the model type to be brought back to top-level design. The return type can be lef, flexllm, ilm, or pnr. *Default*: lef *Data_type*: enum, optional
`-tag` *tag_name*	Allows you to explicitly specify the tag name of the cell pnr model that will be loaded. If a tag is not specified, the `update_module_model` command loads the pnr model of the same tag with the currently loaded design (r`estore_module_model`). For example, if the currently loaded design has tag *init*, then the `update_module_model` command will load the pnr model of tag *init* for the specified cell. It will display an error if the tag is not available. *Data_type*: string, optional

Example

- The following command opens an independent session in batch mode for the full DB of the cell tdsp_core, tag preCts:

```
update_module_model -cell tdsp_core -return_type ilm -plugin_tcl {
 route_eco
 write_module_model
}
```

Related Information

- `write_module_model`
- `commit_module_model`
- `read_module_model`
- `write_partitions`
- `set_module_model`
- Hierarchical and Prototyping Flow chapter in the *Innovus Stylus Common UI User Guide*
  - Integrated Hierarchical Database Flow

write_module_model

```
write_module_model
[-help]
[-allow_port_mismatch]
[-cell string]
[-phys_hier]
[-tag tag_name]
[-type {1801 boundary_model etm flexIlm ilm lef mmmc pnr pnr_shell spef scan_info latency
timing_context extraction_context pin_assign}]
```

Creates a module model. It honors the `set_module_model -default_create_options` command settings.

Parameters

`-allow_port_mismatch`	Creates a new updated LEF. It provides an ability to bring in an updated model with mis-matched ports to top-level design. **Note:** This parameter can be specified with the `lef` type (`-type`) of module models. *Data_type*: bool, optional
`-cell string`	Specifies the cell to save. By default, current top cell will be saved. **Note:** For 2D designs, the `-cell` parameter should be used for pnr type of module model only. **Note**: For 3D IC designs, the `-cell` parameter can be used to specify a list of partition names or design name to save for pnr, lef, or pre_pnr types of module models. *Data_type*: string, optional
`-help`	Outputs a brief description that includes the type and default information for each `write_module_model` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man write_module_model`
`-phys_hier`	Creates the whole physical hierarchy. *Data_type*: bool, optional

`-tag` *tag_name*	Specifies tag name of the saved model. **Note:** You cannot use 'latest' as a tag name as it is a reserved keyword for the latest tag. • If a model is created from an Innovus session that is opened by the `update_module_model` command, then the same tag name of the current loaded design will be used if this option is not specified. • This option is required if a block-level design has been opened explicitly. You only need to specify this tag option for one model type. For subsequently created model types, the same tag can be used if it has not been specified. • If the tag information is not available, the `write_module_model` command displays an error message. *Data_type*: string, optional
`-type {1801 boundary_model etm flexIlm ilm lef mmmc pnr pnr_shell spef scan_info latency timing_context extraction_context pin_assign}`	
	Specifies the type(s) of module model. The following is the list of the supported model types: • `1801`: UPF for power domain • `boundary_model`: Boundary model for timing analysis • `etm`: Extraction timing model • `extraction_context`: Extraction context model • `flexilm`: Flexible interface logic model • `ilm`: Interface logic model • `latency`: Clock latency information • `lef`: LEF abstract • `mmmc`: Multi-mode multi-corner definition • `pin_assign`: SPEF file from RC extraction • `pnr`: Innovus DB • `pnrshell`: Reduced pnr model where only the first level of logics is kept. • `spef`: SPEF file from RC extraction • `scan_info`: Scan chain information • `timing_context`: Timing context model

	Note: Use the `-type 1801` to save UPF associated with an ILM model so that you do not have to keep a track where is the corresponding UPF of an ilm. **Note:** You can specify a list of model types to create multiple types of module models at a time. **Note:** The `pnr_shell` model type is similar to the `flexilm` model but it only has first level of logics. **Note:** The `write_module_model -type spef` honors the `set_module_model -active_rc_corners` command to output SPEF(s) for the specified RC corners. By default, multi corner SPEFs are saved as gzip. The output SPEF(s) are saved at <SHDB_root>/<cell>/<tag>/spef/<topCell>.<cornerName>.spef.gz **Note:** If the `-rc_corner` parameter of the `write_parasitics` command is not specified, the name of the SPEF output file is <SHDB_root>/<cell>/<tag>/spef/<topCell>.spef.gz **Note:** You can use the `timing_context` or `extraction_context` add on types for creating a timing or extraction context model, respectively, for a specific cell or a master/clone instance. This context model can be loaded additionally with the design later in the flow. Default: `pnr` and `lef` *Data_type*: enum, optional

Example

- The following command creates ilm type of module model that has the postRoute tag.
  ```
  write_module_model -type ilm -tag postRoute
  ```

Related Information

- `commit_module_model`
- `check_module_model`
- `read_module_model`
- `write_partitions`
- `set_module_model`
- `update_module_model`
- Hierarchical and Prototyping Flow chapter in the *Innovus Stylus Common UI User Guide*
  - Integrated Hierarchical Database Flow

write_module_model_context

```
write_module_model_context
[-help]
{-cell cellName | -master_clone_inst master_clone_inst}
[-tag tag_name]
[-type {extraction_context timing_context}]
[-type_specific_options type_specific_options]
```

Creates a context model for a specific cell or a master/clone instance. This context model can be loaded additionally with its design later on in the flow.

Parameters

`-cell cellName`	Specifies the cell name(s) to be loaded. You can specify a list of cell names. *Data_type*: string, required
`-help`	Outputs a brief description that includes the type and default information for each `write_module_model_context` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man write_module_model_context`
`-master_clone_inst instance_name`	Specifies the instance name of a master/clone module. Use this option if the instance has clone(s). *Data_type*: string, optional
`-tag tag_name`	Specifies the output tag name. *Default*: latest *Data_type*: string, optional
`-type {extraction_context timing_context}`	Specifies the type of the generated context model. • `extraction_context`: Writes the extraction context information of specified modules/instances. • `timing_context`: Writes the timing context information of specified modules/instances. *Data_type*: enum, optional

`-type_specific_options`	Specifies the options of the type model. • For **`extraction_context`** type of model, the type specific options available are: `{-extension` *`distance`* `| -surrounding_tag` *`name`* `| -ground` *`name`*`}` ◦ `-extension` *`distance`* Specifies the block halo distance (in microns) where routing data will be included in an output DEF file. *Default*: 10 ◦ `-surrounding_tag` *`name`* Specifies the surrounding object tag name. *Default*: Tag name of the current loaded design. ◦ `-ground` *`name`* Specifies the ground net that the surrounding nets will be tied to. • For **`timing_context`** type of model, the type specific options available are: `{-skip_interface_paths	-overwrite	-with_constraints}` ◦ `-skip_interface_paths` Specify this option if the module is instantiated many times at the full-chip top design. It preserves the context data impacting only internal reg-reg paths. During analysis with this context data, interface paths will be unconstrained. ◦ `-overwrite` Specify this option to overwrite the existing `timing_context` model. It overwrites the existing context data. ◦ `-with_constraints` Specify this option to write the SDC file of each of the specified module inside the corresponding sub directories. This option is supported only in SMSC flow and for single instantiated modules. *Data_type*: string, optional

Example

- The following command creates an extraction context model for cell ptn_wrapper with routing data within 50 micron halo distance from ptn_wrapper LEF boundary:

```
read_module_model dtmf_recvr_core -tag init

# Ensure that routing to be cut around specified object should be a LEF
set_module_model -cell ptn_wrapper -type lef -tag my_extract_v1
commit_module_model -mmmc_file design/dtmf_recvr_core.enc.dat/viewDefinition.tcl

route_early_global

write_module_model_context -cell ptn_wrapper -tag my_extract_v1 -type_specific_options {-extension
50 -ground vss1 init} -type extraction_context
```

Related Information

- write_module_model
- commit_module_model
- read_module_model
- write_partitions
- update_module_model
- Hierarchical and Prototyping Flow chapter in the *Innovus Stylus Common UI User Guide*
    - Integrated Hierarchical Database Flow

37

Timing Analysis Commands and Attributes

- check_ldb_version
- check_timing
- check_timing_library
- check_timing_library_consistency
- create_analysis_view
- create_constraint_mode
- create_delay_corner
- create_library_set
- create_opcond
- create_rc_corner
- create_timing_condition
- delete_analysis_view
- get_capacitance_unit
- get_clock_network_objects
- get_constant
- get_equivalent_cells
- get_interactive_constraint_modes
- get_propagated_clock
- get_resistance_unit
- get_socv_reporting_nsigma_multiplier
- get_time_unit
- gui_show_timing_map
- merge_timing_reports

- read_path_descriptions
- read_sdf
- read_transition_file
- read_twf
- report_analysis_coverage
- report_analysis_summary
- report_analysis_views
- report_annotated_assertions
- report_annotated_check
- report_annotated_delay
- report_annotated_parasitics
- report_aocv_derate
- report_case_analysis
- report_cell_instance_timing
- report_clock_gating_check
- report_clock_propagation
- report_clock_timing
- report_clocks
- report_constraint
- report_cppr
- report_design
- report_fanin
- report_fanout
- report_inactive_arcs
- report_instance_library
- report_min_pulse_width
- report_mode
- report_nets

- report_path_based_aocv_derate
- report_path_exceptions
- report_path_groups
- report_ports
- report_socv_interconnect_variation
- report_socv_library
- report_statistical_timing_derate_factors
- report_timing
- report_timing_derate
- report_timing_summary
- report_wire_load
- reset_instance_library
- reset_path_adjust
- reset_path_adjust_group
- reset_sdf_assertions
- reset_timing_derate
- reset_wire_load_mode
- reset_wire_load_model
- reset_wire_load_selection_group
- set_analysis_view
- set_aocv_thresholds
- set_default_view
- set_instance_library
- set_interactive_constraint_modes
- set_io_thresholds
- set_path_adjust
- set_path_adjust_group
- set_socv_rc_variation_factor

- set_socv_reporting_nsigma_multiplier
- set_table_style
- set_timing_derate
- set_wire_load_mode
- set_wire_load_model
- set_wire_load_selection_group
- time_design
- time_design_signoff
- timing Category Attributes
- update_analysis_view
- update_constraint_mode
- update_delay_corner
- update_io_latency
- update_library_set
- update_opcond
- update_rc_corner
- update_timing_condition
- write_combinational_loop_break_constraints
- write_global_slack_report
- write_ldb
- write_path_descriptions
- write_sdc
- write_sdf
- write_timing_windows
- set_socv_constraint_config
- get_socv_rc_variation_factor
- create_property_alias
- get_clock_to_clock_association

- report_lib_arcs
- list_libraries

Path Exception Priorities

The following are the path exception priorities if a path in the design matches more than one path exception:

1. `set_false_path`
2. `set_min_delay`, `set_max_delay`
3. `set_multicycle_path`

If there is more than one exception of a given type, for example the `set_multicycle_path` command, the path exception that is more specific has higher priority. A path exception has higher priority if it specifies more of the `-from`/`-through`/`-to` parameters. For example, the `-from -to` options would have priority over the `-from` option.

If the path has the same number of reference points:

- `-from` option has priority over the `-to` option
- `-to` option has priority over the `-through` option
- `-from` option has priority over the `-to` option

Note: To check for ignored path exceptions, use the `report_path_exceptions -ignored` command.

The following table shows the priorities for path exceptions applied to the same path.

Table 38-1 **Path Exception Priorities**

Priority	Path Exception
1. (Highest)	`set_false_path`
2.	`set_max_delay -from` *`pin_list`*
3.	`set_max_delay -to` *`pin_list`*
4.	`set_max_delay -through` *`pin_list`*
5.	`set_max_delay -from` *`clkwave_name`*

6.	`set_max_delay -to clkwave_name`
7.	`set_max_delay` (The most constraining adjustment has the higher priority over less constraining adjustments.)
8.	`set_multicycle_path -from pin_list`
9.	`set_multicycle_path -to pin_list`
10.	`set_multicycle_path -through pin_list`
11.	`set_multicycle_path -from clkwave_name`
12.	`set_multicycle_path -to clkwave_name`
13. (Lowest)	`set_multicycle_path` (The most constraining adjustment has the higher priority over less constraining adjustments.)

Examples of Path Exception Priorities

The following are examples of the path exception priorities for a path starting at `A`, going through `B`, and ending with `C`. The pairs of examples are ordered such that the highest priority constraint in one pair has precedence over the highest priority constraint in the next pair.

- In the following pair of constraints, the false path overrides the cycle addition for the path from `A` through `B` to `C`. For all other paths from `A`, the cycle addition applies.

  ```
  set_false_path -from A -through B -to C set_multicycle_path -from A 2
  ```

- In the following pair, the first constraint is applied (two cycles added) because the `-from` option has priority over the `-to` option:

  ```
  set_multicycle_path -from A 3 set_multicycle_path -to C 2
  ```

- In the following pair, the first constraint is applied (two cycles added) because it specifies a `-through` pin as well as a `-to` pin:

  ```
  set_multicycle_path -through A -to B 3 set_multicycle_path -to B 2
  ```

- In the following pair, the first constraint is applied (two cycles added) because the `-from` option has precedence over the `-to` option:

  ```
  set_multicycle_path -from CLKA 3 set_multicycle_path -to CLKB 2
  ```

- In the following pair, the second constraint is applied because the second constraint overwrites

the first one:

```
set_multicycle_path -from A 2 set_multicycle_path -from A 3
```

Bidirectional Pin Defaults

Bidirectional pins have both an input pin and an output pin type. This section explains the default pin type that the system uses for bidirectional pins.

Defaults for Pin (or Port) Constraints

The following defines the pin and port types that are used in the next table.

Table 38-2 **Bidirectional Pin (or Port) Type Definitions**
Bidirectional Pin (or Port) Type Definitions

Pin Type	Type 1	Type 2
Blackbox pins	out	in
Instance pins	out	in
Netlist (primary) ports	in	out
Hierarchical ports	in	out

Table 38-3 **Bidirectional Pin Type Defaults**

Pin or Port Constraint	Type	Hierarchical Port
set_input_delay	1	ignored
set_output_delay	2	ignored
set_clock_latency	1	ignored
create_generated_clock	1	used
set_input_transition	1	ignored
set_driving_cell	1	ignored

set_drive	1	ignored

Defaults for Port Constraints

The following table defines the port types that are used in the next table.

Table 10-4 **Bidirectional Port Type Definitions**

Pin Type	Type 1	Type 2
Netlist (primary) ports	in	out
Hierarchical ports	in	out

Table 10-5 **Bidirectional Port Type Defaults**

Port Constraint	Type	Hierarchical Port
set_fanout_load	2	ignored
set_load	2	ignored
set_max_transition	2	ignored

check_ldb_version

```
check_ldb_version
-library string
[-help]
```

Checks the library database version. If the library version is not updated, the details of missing features and recommendations for recompiling the library database are displayed.

Parameters

-help	Prints out the command usage.

`-library` *string*	Specifies the library database (LDB) file whose version is to be printed.

check_timing

```
check_timing
[-help]
[-check_only warning_list]
[-exclude_warnings warning_list]
[-include_warnings warning_list]
[-tcl_list]
[-type type_list]
[-verbose]
[-view view_name]
[> <file_name[.gz]>]
[>> <file_name[.gz]>]
[ port_or_pin_list]
```

Performs a variety of consistency and completeness checks on the timing constraints specified for a design. Valid types are: `clocks`, `clock_clipping`, `constant_collision`, `endpoints`, `input`, and `loops`.

The checks include arrival time and external delay (or required time) for each clock in a multiple clock system. In addition, clock connectivity and data connectivity are checked to make sure the clock or data is propagated as expected. Clock gating points are also reported.

Use this command on generic or mapped netlists. The `check_timing` command considers constants applied to a pin while reporting warnings.

> Use the `check_timing` command before using any commands for timing analysis. Warnings are displayed when problems occur. The warnings are listed in the table below. For best results, rerun the `check_timing` command after resolving each warning. Redirecting the output to a file is the quickest run method.

Parameters

`-check_only` *warning_list*	Allows you to specify warning types. For example, the following command lists the specified warnings only: check_timing -check_only {uncons_endpoint clock_crossing} This list can also include warnings that are not listed by default. Valid entries in *warning_list* are: clock_clipping_freq, clock_clipping_gate, clock_crossing, clock_expected, clock_gating_controlling_edge_unknown, clock_gating_no_data, clock_missing_at_sequential_output, clock_not_propagated, clocks_masked_by_another_clock, const_collision, data_check_multiple_reference_signal, data_check_no_reference_signal, ideal_clock_waveform, invalid_pll_configuration, loop, master_clk_edge_not_reaching, missing_clock_groups, missing_drive, multiple_clock, no_drive, no_gen_clock_source, no_input_delay, partial_input_delay, pulse_non-pulse_clock_merge, signal_level, uncons_endpoint, uncons_endpoints_due_to_case_analysis

{> | >>} *file_name*	When specified with >, writes the report to the specified file name. If the file already exists, the software overwrites it. When specified with >>, writes the report to the specified file name. If the file already exists, the software concatenates the report to the end of the file. The *filename* parameter must be the last argument in the list. The *filename* and -tcl_list parameters are mutually exclusive; you cannot specify them together. **Note:** To write a compressed report, add the .gz extension to the file name. *Default:* Report is displayed on standard output without being saved.
-exclude_warnings *warning_list*	Excludes the specified warning types when generating the report. Use this parameter to exclude any of the warnings included in the report by default. By default, the software includes the following warnings in the report: clock_expected, loop, ideal_clock_waveform, master_clk_edge_not_reaching, partial_input_delay, missing_drive, no_input_delay, no_drive, no_gen_clock_source, uncons_endpoint. The complete list of warnings and their descriptions is given in Table 10-6.

`-include_warnings` *warning_list*	Includes the specified warning types when generating the report. Use this parameter to add any of the warnings not included in the report by default. By default, the software reports all warnings except the following: `clock_clipping_gate`, `const_collision`, `clock_crossing`, `clock_not_propagated`, `missing_drive`, `multiple_clock`, `data_check_multiple_reference_signal`, `data_check_no_reference_signal`, `clock_gating_no_data`, `clock_gating_controlling_edge_unknown`, `clock_missing_at_sequential_output`, `signal_level`, and `clocks_masked_by_another_clock`. The complete list of warnings and their descriptions is given in Table 10-6.
port_or_pin_list	Lists pins or ports. For example: `check_timing -verbose [get_ports in*]`
`-tcl_list`	Produces the report in Tcl list format instead of a tabular format. This is useful for integrating timing with custom Tcl functions, and also for customizing report generation. The `-tcl_list` and *filename* parameters are mutually exclusive; you cannot specify them together.
`-type` *type_list*	Limits the checks to the specified type. Valid types are: `clocks`, `endpoints`, `inputs`, `nets`, `loops`, `clock_clipping`, `clock_gating_inferred`, and `constant_collision`. *Default*: All types. Only check violations will be reported. To report all the checks you can use the `timing_check_timing_report_all_checks` attribute. When set to `true`, the software reports all the checks performed by the `check_timing` command, even if there are no violations. By default, this attribute is set to `false`.

	`clock_clipping`: Warns when there is a problem with clock gating that can result in clock clipping. Reports the following warnings: • clock_clipping_freq • clock_clipping_gate
	`clocks`: Reports the following clock warnings: • clock_crossing • clock_expected • clock_not_propagated • ideal_clock_waveform • master_clk_edge_not_reaching • no_gen_clock_source
	`clock_gating_inferred`: Warns when inferred clock gating checks are not analyzed by the software, even if a clock signal reaches and is used downstream the clock tree. Reports the following warnings. • clock_gating_controlling_edge_unknown • clock_gating_no_data
	`constant_collision`: Warns when there is a constant collision on a net connected to the pin or if there is a contradiction on a pin. Constant collision occurs when a net is driven simultaneously by conflicting values and a constant contradiction occurs when a constant constraint on a pin conflicts with the driven value.

	`endpoints`: For output ports, warns if no output delay constraint is applied to the port or if a specific (min/max rise/fall) output delay is missing. It also warns if an output delay constraint is applied with no clock information. For all endpoints (output ports and register data pins), it warns if any incoming signal is unconstrained. Also warns if a clock arrives where clock is not expected, or if multiple signals arrive at an endpoint.
	`inputs`: Reports the following input delay, arrival time assertion, and drive assertion warnings: • partial_input_delay • missing_drive • no_input_delay • no_drive
	`loops`: Warns when a combinational loop is detected. Reports the loop and the arc used to break the loop. The `-pin` option is ignored in this case. Also reports loops caused by generated clocks.
`-verbose`	Gives detailed information about the warnings. When specified, warnings are grouped and sorted by warning type. *Default:* Gives a summary of how many warnings of each type exists.
`-view`	Reports warning(s) that belong to the specified view.

Command Order

Use this command after setting all the constraints but before optimizing your design or generating any

timing reports.

Warning Messages

The following table shows the `check_timing` warning messages.

check_timing Warning Messages

Warning	Description
`clock_clipping_gate`	**Warning**: Clock clipping possible due to wrong gate type or wrong trigger for data input.
`clock_clipping_freq`	**Warning**: Clock clipping possible due to incompatible clock signal and data signal frequencies.

`clock_crossing`	**Warning**: Clock domains interact. **Comments:** Checks clock interactions between multiple clock domains in the design. When you specify this parameter, the Timing Check Crossing Clocks report shows the interaction of clocks in the design and indicates whether the paths between the clocks are `False` or `Partial False`. If a clock launches one or more paths, which are captured by other clocks, the report lists all pairs of crossing clocks. • This includes inferred clock gating with different reference and signal clock. • If a path has been disabled using the `set_disable_timing` command, crossing clocks on that path are not reported. If all the paths between two clocks are false paths or they are exclusive/asynchronous clocks, the path is marked as `False`. The false path can be defined on clocks, for example, `set_false_path -clock_to CLK` or on pins or ports, such as, `-through buf1/Y`. If only part of paths are set as false paths or exclusive/asynchronous clocks, the path is marked as `Partial False`.
	• If a false path is defined only for one check type (say, setup), the crossing clocks are marked as `Partial False` if the simultaneous setup hold mode is set to `on`. The crossing clocks are marked `False` if the analysis type is setup and the simultaneous setup hold mode is set to `off`. • If only one edge (rise/fall) of a particular clock pair has false paths (say, using `-clock_fall`), then the path is marked as a `Partial False`.

`clock_expected`	**Warning:** Clock not found where clock is expected. **Comments:** Indicates that no clock signal is defined at the clock pin. See create_clock.
`clock_gating_controlling_edge_unknown`	**Warning:** Unknown controlling edge to perform clock gating check. **Comments:** Reports library cells with complex functions (MUX or AOI) where controlling edge of signal to perform clock gating check cannot be determined automatically. You need to specify correct edge by `-high`/`-low` option using the `set_clock_gating_check` command to resolve such warning. **Note:** The software does not report this warning when either: • `set_disable_clock_gating_check` command is applied on such pins, or • global settings to disable clock gating checks are on.

`clock_gating_no_data`	**Warning:** No data signal reaching clock gating enable pin **Comments:** Reports clock gating enable pins where no valid data signal arrives. You need to do either of the following: • specify valid data signal to reach enable pin of clock gating cells, or • remove any path exceptions or disabled checks that blocks valid data signal to reach enable pin **Note:** The software does not report this warning when either • `set_disable_clock_gating_check` is applied on such pins, or • global settings to disable clock gating checks are on.
`clocks_masked_by_another_clock`	**Warning:** Clock(s) in fanin cone are dropped due to another clock **Comments:** Reported timing pins on which creation of a regular or a generated clock causes clock(s) in fanins of such pins have been blocked or masked from propagating further. However, for pins having only master clock waveform on the fanin, this warning is not reported. You can use the detailed mode (`-verbose` option) to report names of all the incoming clock waveforms that have been blocked.

`clock_missing_at_sequential_output`	**Warning:** Found latency phase at sequential output on clock tree instead of clock phase **Comments:** Reports output pins of sequential elements that lie on source clock tree for the generated clocks created downstream and do not have any valid clock phase. This indicates possible missing clock related checks (e.g., clock gating) on downstream clock tree. You can add clocks on this pin or downstream clock tree only if no valid clock exists until downstream generated clock creation point.
`clock_not_propagated`	**Warning:** Clock not propagated **Comments:** Reports ideal clocks reaching the reference pin of a timing check.
`const_collision`	**Warning:** Constant collision **Comments:** See `set_case_analysis`.
`data_check_multiple_reference_signal`	**Warning:** Multiple clocked signals are reaching data check reference pin **Comments:** Reports on reference pins for non sequential timing check (data check) where more than one reference data arrives.
`data_check_no_reference_signal`	**Warning:** No clocked signal is reaching data check reference pin **Comments:** Reports reference pins for non-sequential timing check (data check) where no valid reference signal (data phase) arrives. You can do one of the following to resolve such warnings: • add valid reference signals to reach at such reference pins • remove path exceptions or disabled arcs that are blocking such reference signals

`ideal_clock_waveform`	**Warning:** Clock waveform is ideal. **Comments:** Reports clocks that are ideal and are not propagated. This warning type is reported by default (unless `-exclude_warning` or `-check_only` parameter is specified). You can use the `set_propagated_clock` command to avoid this warning.
`invalid_pll_configuration`	**Warning:** No valid path exists from output pin to feedback pin of PLL. **Comments**: Shows PLL (phase locked loop) check warnings. These checks allow you to identify and fix issues in the PLL flow.
`latch_fanout_clock_connectivity`	**Warning**: Latch and its fanout latch work on same clock. **Comments**: Checks for a latch and its fanout latch(es) that works on the same clock.
`loop`	**Warning:** Timing loop found in the design **Comments:** Indicates that there are timing loops in the design. It also includes loops caused by generated clocks.
master_clk_edge_not_reaching	**Warning:** Master clock edge does not reach the generated clock target. **Comments:** This warning means that rise/fall latency edge is not reaching to create generated clock root from the master clock. You can verify if the path that is seeing this warning is actually broken, or edge is somewhere disabled earlier on this path. There is a global `timing_enable_genclk_edge_based_source_latency` that controls how the software chooses generated clock source latency paths.

`missing_clock_groups`	**Warning**: Set clock group related checking. **Comments:** Reports any potential issues with the defined clock grouping data.
`partial_input_delay`	**Warning:** Indicates that a rise or fall input delay assertion is not applied.
`missing_drive`	**Warning:** Missing specific (rise/fall) drive constraint **Comments:** Either a rise or fall drive constraint is applied and the other one is missing.
`multiple_clock`	**Warning:** Multiple clocks are reaching clock reference pin. **Comments:** Reports clock pins of sequential or clock gating cells where more than one clock waveform arrives. This warning is for information only and the software performs timing analysis with respect to all such clocks.
`no_input_delay`	**Warning:** No input delay assertion with or without `-clock` option specified. **Comments:** See `set_input_delay`.
`no_drive`	**Warning:** No drive constraint **Comments:** See `set_driving_cell` and `set_drive`.
`no_gen_clock_source`	**Warning:** No clock source found for generated clock. **Comments:** Indicates that the source pin of the clock generated by the `create_clock` command is not driven by a clock source.
`pulse_non-pulse_clock_merge`	**Warning**: Merge of pulse and non-pulse clock paths. **Comments**: Reports points where a pulse clock and a non-pulse clock merges.

`signal_level`	**Warning:** Driver pin voltage value do not match with sink pin voltage level **Comments:** Reports net driver/receiver pin pairs for which the pin voltage does not match.
`uncons_endpoint`	**Warning:** No constrained signal reaching pin. **Comments:** For output ports: Primary output port has no implicit timing check. Use `set_output_delay`, `set_max_delay`, or `set_min_delay` to create a valid constraint. For register data pins: There is no valid clock reaching the CK pin of the register data pin. Check for missing `create_clock` for the clock network. The endpoints are reported as unconstrained in both in early and late timing. This does not cover all the cases and scenarios because of which endpoint can become unconstrained. To cover all possible cases, you can use the `report_analysis_coverage` command.
`uncons_endpoints_due_to_case_analysis`	**Warning:** Unconstrained signal arriving at end point due to case analysis **Comments:** Reports unconstrained endpoints due to case analysis. To enable this warning, you must set the `timing_report_check_timing_unconstrained_endpoints_due_to_constants` attribute to `true`.

Examples

- Consider the following example:

```
in---------|>-----------|>---------out

           buf1 buf2
```

The Timing Check Crossing Clocks reports `False` or `Partial False` in some of the scenarios given below:

- set_input_delay with respect to CLK1 on in and set_output_delay with respect to CLK2 on out:

```
From Clock         Crossing Clocks
-------------------------------------------
CLK1               CLK2
```

- `set_input_delay` with respect to `CLK1` on `in`, `set_output_delay` with respect to `CLK2` on `out`, and `set_false_path -to CLK2` (clock-based false path):

```
From Clock         Crossing Clocks
-------------------------------------------
CLK1               CLK2   (False)
```

- If the `set_false_path` becomes `set_false_path -rise_to CLK2`, the following is reported:

```
From Clock         Crossing Clocks
-------------------------------------------
CLK1               CLK2   (Partial False)
```

- The following command displays the default summary report:

```
check_timing
-------------------------------------------------------------------------------------------
--------------
                                    TIMING CHECK SUMMARY
-------------------------------------------------------------------------------------------
--------------
Warning                          Warning Description
      Number of

     Warnings
-------------------------------------------------------------------------------------------
--------------
master_clk_edge_not_reaching    Master clock edge does not reach the generated clock
target 2
no_drive                        No drive assertion
       4
no_input_delay                  No input delay assertion with respect to clock
```

```
       4
uncons_endpoint                 Unconstrained signal arriving at end point
       4
-------------------------------------------------------------------------------
-------------
```

- The following command displays a specified type of warning using the `-type` option if a timing loop is found in the design:
  check_timing -type loops

```
---------------------------------------
        TIMING CHECK SUMMARY
---------------------------------------
Warning Warning Description Number of
Warnings
---------------------------------------
loop Timing loop found in the design 0
---------------------------------------
```

- The following command provides more information about problems and prints a summary using the `-verbose` option:
  check_timing -verbose

```
-------------------------------------------------------------------------------
-------------
                                TIMING CHECK SUMMARY
-------------------------------------------------------------------------------
-------------
Warning                       Warning Description
     Number of

     Warnings
-------------------------------------------------------------------------------
--------------
master_clk_edge_not_reaching  Master clock edge does not reach the generated clock
target  2
no_drive                      No drive assertion
      4
```

```
no_input_delay                No input delay assertion with respect to clock
      4
uncons_endpoint               Unconstrained signal arriving at end point
      4
--------------------------------------------------------------------------------
---------------

                                TIMING CHECK DETAIL
--------------------------------------------------------------------------------
---------------
Pin                            Warning
--------------------------------------------------------------------------------
---------------
u2/Q    Master clock edge does not reach the target for generated clock
CLK_W_4_gen1_inv
CLK4    No drive assertion
IN4     No input delay assertion with respect to clock
IN3     No input delay assertion with respect to clock
OUT4    Unconstrained signal arriving at end point
OUT3    Unconstrained signal arriving at end point
--------------------------------------------------------------------------------
---------------
```

- When `timing_report_check_timing_unconstrained_endpoints_due_to_constants` attribute is enabled, the case analysis warnings are reported as shown below:

```
tempus 11> check_timing
--------------------------------------------------------------------------------

                                TIMING CHECK SUMMARY
--------------------------------------------------------------------------------
Warning              Warning Description                                Number of
                                                                        Warnings
--------------------------------------------------------------------------------
clock_expected       Clock not found where clock is expected              4202
no_drive             No drive assertion                                   241
no_input_delay       No input delay assertion with respect to clock       160
```

```
uncons_endpoint      Unconstrained signal arriving at end point         145259
uncons_endpoints_due_to_case_analysis
                  >>>>Unconstrained signal arriving at end point due to case analysis
                                                                        5969437
-------------------------------------------------------------------------------
```

Related Commands

- report_timing

check_timing_library

```
check_timing_library
[-help]
[-check_power]
[-cell string]
[-library string]
[-out_file string]
[-report_missing_power_only]
```

Checks the contents of the timing library and reports any inconsistencies in the log file.

You can use the `check_timing_library` command after importing the design and timing libraries.

Parameters

`-help`	Prints out the command usage.
`-check_power`	Checks if power constructs, such as, internal_power, leakage_power, or ecsm_power, are defined for the cells. **Note**: The `-check_power` parameter is optional. However, if you specify `-report_missing_power_only`, then you must specify `-check_power`. If you use the `-library` parameter, you do not have to specify the `-check_power` parameter, because when a library-specific cell is given as input, the software automatically checks for power constructs. *Default:* Only checks if timing is defined for the cells.
`-cell string`	Specifies the cell name.

`-library` *string*	Specifies the timing library in which to check a particular cell. **Note:** If you use the `-library` parameter, you do not have to specify the `-check_power` parameter, because when a library-specific cell is given as input, the software automatically checks for power constructs.
`-out_file` *string*	Saves the report output to the specified file.
`-report_missing_power_only`	Reports only those cells that do not have any defined internal_power (at the pin level), or leakage_power. You must specify the `-check_power` parameter with the `-report_missing_power_only` parameter.

Examples

- The following command checks if timing is defined for all cells that have been read into the design, and writes a report to the log file:
  ```
  check_timing_library
  ```
- The following command checks if timing is defined for all cells that have been read into the design, and writes a report to a file named `CheckTiming1`:
  ```
  check_timing_library -out_file CheckTiming1
  ```
- The following command checks if timing and power constructs are defined for all cells that have been read into the design, and writes the report to the log file:
  ```
  check_timing_library -check_power
  ```
- The following command checks if timing and power constructs are defined for all cell that have been read into the design, and writes the report to a file named `CheckTiming2`:
  ```
  check_timing_library -out_file CheckTiming2 -check_power
  ```
- The following command checks if the internal_power and/or leakage_power constructs are defined for all cells that have been read into the design, and reports only those cells that do not have the constructs defined. The software writes the report to a file named `MissingPower`:
  ```
  check_timing_library -out_file MissingPower -check_power -report_missing_power_only
  ```

check_timing_library_consistency

```
check_timing_library_consistency
```

Checks for missing libraries for instances in the active view.

This command checks whether the library for the cell associated with the instance is present in the associated power domain library set of the instance in the active views.

Parameters

None

Examples

- The following command displays a report showing missing libraries:

  ```
  check_timing_library_consistency
  ```

  The software displays the following information:

  ```
  Checking the Library binding for instance in active view 'vA'
  Pass. All instances have library definition in view 'vA'
  Checking the Library binding for instance in active view 'vB'
  No library found for instance 'U1', cell 'AND2X1', in powerdomain 'PD_BAD', of view
  'vB'
  Fail. Missing Library for some instance found in view 'vB'
  Checking the Library binding for instance in active view 'vAB'
  Pass. All instances have library definition in view 'vAB'
  ```

create_analysis_view

```
create_analysis_view
-constraint_mode modeName
-delay_corner dcCornerObj
-name viewName
```

Creates an analysis view object that associates a delay calculation corner with a constraint mode. An analysis view object provides all of the information necessary to control a single multi-mode multi-corner analysis.

After creating analysis views, use the `set_analysis_view` command to specify which views to use for setup and hold optimization or timing analysis.

Use this command after creating constraint modes (`create_constraint_mode`) and delay calculation corners (`create_delay_corner`).

Parameters

`-name` *viewName*	Specifies the name of the analysis view being defined.
`-constraint_mode` *modeName*	Specifies the name of the constraint mode to associate with this analysis view. A constraint mode groups a set of constraints to be used with any given analysis. To create the constraint mode, use the `create_constraint_mode` command.
`-delay_corner` *dcCornerObj*	Specifies the name of the delay calculation corner object to associate with this analysis view. A delay calculation corner object contains all of the information needed to control delay calculation for a specific analysis view. To create a delay calculation corner, use the `create_delay_corner` command.

Examples

- The following command creates an analysis view called `missionSlow` using the constraint mode `missionSetup` and the delay calculation corner `dcWCCOM`:

```
create_analysis_view
-name missionSlow
-constraint_mode missionSetup
-delay_corner dcWCCOM
```

- The following command creates an analysis view called `missionFast` using the constraint mode `missionHold` and the delay calculation corner `dcBCCOM`:

```
create_analysis_view
-name missionFast
-constraint_mode missionHold
-delay_corner dcBCCOM
```

Related Topics

- Design Import and Export in Stylus chapter in the *Innovus System User Guide.*
- `set_analysis_view`
- `create_constraint_mode`
- `create_delay_corner`

create_constraint_mode

```
create_constraint_mode
-name modeName
-sdc_files {file1.sdc file2.sdc ...}
[-ilm_sdc_files {file1.sdc file2.sdc ...}]
[-tcl_variables {{var_name1 value1} {var_name2 value2} {var_name3 value3} ...}]
```

Associates a list of SDC constraint files with a specified constraint mode name, for multi-mode multi-corner analysis. This constraint mode name can be referred to later when creating analysis views. A constraint mode defines one of possibly many different functional, test, or Dynamic Voltage and Frequency Scaling (DVFS) modes of a design. SDC files can be shared by many different constraint modes, and the same constraint mode can be associated with multiple analysis views.

SDC files typically contain timing analysis information, such as the clock specifications, case analysis constraints, I/O timings, and path exceptions that make each mode unique. If the ILM is being used, the timing system utilizes the files specified with `-ilm_sdc_files` during the flatten ILM mode (`flatten_ilm`); otherwise, the system uses the files specified with `-sdc_files`.

Parameters

`-ilm_sdc_files {file1.sdc file2.sdc ...}`	Specifies a Tcl list of ILM SDC files to be included in the constraint mode.
`-name modeName`	Specifies the name of the constraint mode to be created.
`-sdc_files {file1.sdc file2.sdc ...}`	Specifies a Tcl list of SDC files to be included in the constraint mode.
`-tcl_variables {{var_name1 value1} {var_name2 value2} {var_name3 value3} ...}`	
	Specifies a list of tcl variables and values to be used in the constraint mode.

Examples

- The following command groups the SDC files `io.sdc`, `mission1-clks.sdc`, and `mission1-except.sdc` to create a mode object named `missionSetup`:

```
create_constraint_mode -name missionSetup

-sdc_files [list io.sdc mission1-clks.sdc mission1-except.sdc]
```

- The following command groups the SDC files `io.sdc`, `mission1-clks.sdc`, `mission1-except.sdc`, and `dont_use.sdc` to create a mode object named `missionHold`:

```
create_constraint_mode -name missionHold

-sdc_files [list io.sdc mission1-clks.sdc mission1-except.sdc      dont_use.sdc]
```

- The following command groups the SDC files `test-io.sdc`, `test-clks.sdc`, and `test-except.sdc` to create a mode object named `testHold`:

```
create_constraint_mode -name testHold

-sdc_files [list test-io.sdc test-clks.sdc test-except.sdc]
```

Related Topics

- Importing and Exporting Designs chapter in the *Innovus System User Guide*
  - Configuring the Setup for Multi-Mode Multi-Corner Analysis
  - Creating Constraint Mode Objects
  - `update_constraint_mode`
  - `create_analysis_view`

create_delay_corner

```
create_delay_corner
[-help]
-name delayCornerName
[-early_estimated_worst_irDrop_factor percent]
[-early_irdrop_data files_or_directories]
[-early_irdrop_files list_of_files]
[-early_rc_corner rcCornerObj]
[-early_temperature_files list_of_files]
[-early_timing_condition conditional_pairs]
[-late_estimated_worst_irDrop_factor percent]
[-late_irdrop_data files_or_directories]
[-late_irdrop_files list_of_files]
[-late_rc_corner rcCornerObj]
[-late_temperature_files list_of_files]
[-late_timing_condition conditional_pairs]
[-irdrop_data files_or_directories]
[-irdrop_files list_of_files]
[-pg_net_voltages voltage_pairs]
[-rc_corner rcCornerObj]
[-si_enabled <true | false>]
[-temperature_files list_of_files]
[-timing_condition conditional_pairs]
```

Creates a named delay calculation corner object that can be referenced later when creating an analysis view. A delay calculation corner provides all of the information necessary to control delay calculation for a specific view. Each corner contains information on the libraries to use, the operating conditions with which the libraries should be accessed, and the RC extraction parameters to use for calculating parasitic data. Delay corner objects can be shared by multiple top-level analysis views.

ⓘ The `-early_*` and `-late_*` parameters should not be used to represent best and worst case corners. Instead use separate delay corners to define Best-Case and Worst-Case differences. Use the -early_* and -late_* parameters within a single delay calculation corner to control on-chip variation. In this latter situation different conditions or libraries will be used for the launch and capture paths of individual timing paths.

Note: A single delay calculation corner object specifies the delay calculation rules for the entire design. If a design includes power domains, the delay calculation corner can contain domain-specific subsections that specify the required operating condition information, and any necessary timing library

rebinding for the power domain. Use the `update_delay_corner` command to add a power domain definition to a delay calculation corner.

Parameters

`-help`	Prints out the command usage.
`-early_estimated_worst_irDrop_factor` *`per cent`*	Specifies the early voltage derating factor. When this parameter is specified, the software performs delay calculations using the voltage = nomVolt * (1 – value). *Default*: 0.0
`-early_irdrop_data` *`files_or_directories`*	Specifies the IR drop files or directories to apply when calculating the early arrival times at a single delay corner.
`-early_irdrop_files` *`list_of_files`*	Specifies the list of IR drop files to apply when calculating early arrival times at a single power domain. This parameter is used primarily when configuring the multi-mode multi-corner environment for on-chip-variation (OCV) analysis. **Note**: This parameter is obsolete and will be removed in a future release. To ensure compatibility with future releases, you should update your scripts.
`-early_rc_corner` *`rcCornerObj`*	Specifies the RC corner object to associate with the early corner object.
`-early_temperature_files` *`list_of_files`*	Specifies the name of the temperature file for temperature aware delay calculation for early corner.

`-early_timing_condition` *conditional_pairs*	Specifies pair list of domains/supply sets and timing conditions to associate with this delay corner object.
`-irdrop_data` *files_or_directories*	Specifies the IR drop files or directories to apply to both the early and late delay calculation for the specified delay corner object.
`-irdrop_files` *list_of_files*	Specifies the list of IR drop files to apply to both early and late delay calculation for this power domain object. This parameter is used primarily to configure single-corner or Best-Case Worst-Case (BC-WC) analysis modes. **Note**: This parameter is obsolete and will be removed in a future release. To ensure compatibility with future releases, you should update your scripts.
`-late_estimated_worst_irDrop_factor` *percent*	Specifies the late voltage derating factor. When this parameter is specified, the software performs delay calculations using the voltage = nomVolt * (1 – value). *Default*: 0.0
`-late_irdrop_data` *files_or_directories*	Specifies the IR drop files or directories to apply when calculating the late arrival times at a single delay corner.
`-late_irdrop_files` *list_of_files*	Specifies the list of IR drop files to apply when calculating late arrival times at a single power domain. This parameter is used primarily when configuring the multi-mode multi-corner environment for on-chip-variation (OCV) analysis. **Note**: This parameter is obsolete and will be removed in a future release. To ensure compatibility with future releases, you should update your scripts.

`-late_temperature_files` *list_of_files*	Specifies the name of the temperature file for temperature aware delay calculation for late corner.
`-late_rc_corner` *rcCornerObj*	Specifies the RC corner object to associate with the late corner object.
`-late_timing_condition` *conditional_pairs*	Specifies pair list of domains/supply sets and timing conditions to associate with this delay corner object. **Note**: The `-early_timing_condition` and `-late_timing_condition` parameters should be specified together.
`-name` *delayCornerName*	Specifies the delay calculation corner to which to add the power domain definition, or to which to update the delay corner or power domain attribute values.
`-rc_corner` *rcCornerObj*	Specifies the new RC corner object to associate with this delay corner.
`-pg_net_voltages` *voltage_pairs*	Specifies the power/ground net voltages to be used for both early/late timing analysis.
`-si_enabled {true | false}`	Controls signal integrity (SI) analysis. When set to `true`, the analysis of all MMMC views will include SI analysis by default. You can use this parameter to further determine which analysis views receive SI analysis and which do not. When this parameter is set to `false` on a delay corner, the software will disable SI analysis for all the views that reference that corner. By default (or when this parameter is set to `true`), SI analysis is computed for all the views related to that corner - whenever SI is globally enabled. (The global control of SI analysis is determined by the `delaycal_enable_si true` attribute setting.)
`-temperature_files` *list_of_files*	Specifies the name of the temperature file for temperature aware delay calculation for single corner.

`-timing_condition` *conditional_pairs*	Specifies pair list of domains/supply sets and timing conditions to associate with this delay corner object.

Examples

- The following example specifies a list of conditional pairs of domains/supply sets and the timing conditions associated with the specified delay corner object:

```
create_delay_corner -name ccworst_setup\
-timing_condition {tc_1 PD1@tc_2 PD2@tc_2 PD3@tc_2 PD4@tc_2}
```

Related Commands

- update_delay_corner

create_library_set

```
create_library_set
[-help]
[-library_side_file string]
-name libSetName
-timing {lib1.lib lib2.lib lib3.lib ...}
[-aocv string]
[-si {lib1.cdb lib2.cdb lib3.udn lib4.udn...}]
[-socv string]
```

Associates a TCL list of timing and cdB/UDN libraries with a specified library set name. Library sets allow a group of library files to be treated as a single entity so that higher-level descriptions (delay calculation corners) can simply refer to the library configuration by name. The same library set can be referenced multiple times by different delay calculation corners.

Note: You must have a set of timing libraries specified in the configuration file for the software to initialize properly. When configuring your multi-mode multi-corner environment, you can choose to specify these libraries in your own custom library set, or refer to them through the two library set definitions that are automatically initialized from the configuration file: `default_libs_min` and `default_libs_max`.

Note: The `create_library_set` command determines the priority of ECSM vs CCS data (if both are present). The software first checks for ECSM construct, if it is missing then CCS construct is loaded.

ⓘ The order in which you define timing libraries is important. The software considers the first library you specify in the list as the master library, with each successive library having a lower priority.

Parameters

`-help`	Prints out the command usage.
`-aocv`	Specifies the aocv libraries to include in the library set.
`-library_side_file string`	Specifies the library side file.

`-name` *libSetName*	Specifies the name for the library set being created. This name is used to associate the library list with a specific delay calculation corner.
`-si {`*lib1*`.cdb` *lib2*`.cdb`*lib3*`.udn` *lib4*`.udn…}`	Specifies cdB libraries and/or user-defined noise (UDN) models to include in the library set. The cdB and/or UDN libraries are required for performing signal integrity analysis. **Note**: You can use this parameter for CCS-N or ECSM-N models, that is, for cdBs only.
`-socv` *string*	Specifies the SOCV libraries to include in the library set.
`-timing {`*lib1*`.lib` *lib2*`.lib` *lib3*`.lib …}`	Specifies the timing libraries to include in the library set. **Note:** Libraries intended for use in an ECSM-based multi-library interpolation must be grouped within an additional level of Tcl list structure to identify the intended usage to the library subsystem. Specify three timing libraries that are characterized at different voltages (typically the lower and upper voltage bounds, and a third intermediate point). All voltages required by the software will be interpolated from this set of libraries. The libraries in the set can differ only in the characterization voltage; they must be identical to each other in all other aspects. However, if you want to perform multi-temperature based library interpolation, you can use two library files that are characterized at different temperatures for each voltage value. See the examples for creating an ECSM-based multi-library set and multi-temperature based library interpolation in the Examples section.

The ECSM and CCS libraries can be very large and loading them may take significant time. You can convert the ASCII NLDM (Liberty .lib), ECSM, and CCS libraries to a binary version using the `write_ldb` command. For conversion of libraries to a binary version, refer to `write_ldb`.

Instead of the original ASCII libraries, you can specify these binary libraries in the configuration file:

- **Original MMMC setup file**:

```
create_library_set -name libName -
timing "AAA.lib BBB.ecsm CCC.ccs"
```

- **New MMMC setup file**:

```
create_library_set -name libName -
timing "AAA.ldb BBB.ldb CCC.ldb"
```

Examples

- The following command creates a library set that associates timing libraries and cdB libraries with a nominal voltage of 1 volt with the library name `IsCOM-1V`:

```
create_library_set
-name IsCOM-1V -timing [list stdcell_F_1V.lib ram_F.lib pad.lib]
-si [list stdcell_F_2.cdb ram_F.cdb pad.cdb]
```

- The following command creates a library set that can be used to specify the 2-volt version of a core cell library, which could be associated later with a 2-volt MSMV power domain specified in the design:

```
create_library_set
-name IsCOM-2V
-timing [list stdcell_F_2V.lib]
```

Related Topics

- Design Import and Export in Stylus chapter in the *Innovus System User Guide.*
- `update_library_set`

create_opcond

```
create_opcond
[-help]
-name opcondName
[-process processValue]
[-temperature temperatureValue]
[-voltage voltageValue]
```

Creates a set of virtual operating conditions in the specified library without actually modifying the library.

Parameters

`-help`	Prints out the command usage.
`-name` *opcondName*	Specifies the name for the operating condition being created.
`-process` *processValue*	Specifies the process value for the operating condition.
`-temperature` *temperatureValue*	Specifies the temperature value for the operating condition.
`-voltage` *voltageValue*	Specifies the voltage value for the operating condition.

Examples

- The following command creates a virtual operating condition called `PVT1`:

```
> create_op_cond -name PVT1
   -process 1.0
   -voltage 1.2
   -temperature 120
```

- The following command creates a virtual operating condition called `PVT2`:

```
> create_op_cond -name PVT2
   -process 1.0
   -voltage 1.2
   -temperature 20
```

Related Commands

- create_delay_corner
- create_timing_condition
- update_timing_condition

create_rc_corner

```
create_rc_corner
-help
[-temperature temperatureValue]
[-cap_table capTableFile]
-name rcCornerName
[-post_route_cap {capFactor1 capFactor2 capFactor3}]
[-post_route_clock_cap {capFactor1 capFactor2 capFactor3}]
[-post_route_clock_res {resFactor1 resFactor2 resFactor3}]
[-post_route_res {resFactor1 resFactor2 resFactor3}]
[-post_route_cross_cap {xcapFactor1 xcapFactor2 xcapFactor3}]
[-pre_route_cap capFactor]
[-pre_route_clock_cap capFactor]
[-pre_route_clock_res resFactor]
[-pre_route_res resFactor]
[-qrc_tech fileName]
[-via_variation_file viaVariationFile]
```

Creates a named RC corner object that can be referenced later when creating a delay calculation corner object. An RC corner object provides the software with all of the information necessary to properly extract, annotate, and use the RCs for delay calculation.

Note: You must use the RC corner scaling attributes when running the software in multi-mode multi-corner analysis mode. Use the extract_rc category attributes to set the extraction mode for all RC corner objects in the design.

Note: In multi-mode multi-corner mode, only the default RC corners created by the software inherit the derating factor values defined in the configuration file.

You can set different post-route scale factors for the post-route engine variants by specifying them as duplets and triplets. For example,

resFactor1 resFactor2 resFactor3

Where:

- Single value: If you specify one value, the scale factor applies to effort level `low`. Scale factor value of `1` is used for `medium` and `high` by default.
- Duplet: If you specify two values, the first value is used for effort level `low` and the second value is used for `medium`. Scale factor value of `1` is used for `high` by default.
- Triplet: If you specify three values, the first value is used for effort level `low`, the second value is

used for `medium`, and the third value is used for `high`.

Parameters

`-help`	Outputs a brief description that includes type and default information for each create_rc_corner parameter. For a detailed description of the command and all of its parameters, use the man command: `man create_rc_corner`.
`-cap_table` *capTableFile*	Specifies the capacitance table to use for extraction when using this RC corner. *Default:* The Innovus System extraction system uses internal rules for extracting resistances and capacitances. *Data_type*: string, optional
`-pre_route_cap` *capFactor*	Specifies the capacitance scale factor for RC Extraction in pre-route mode. *Default:* `1.0` *Data_type*: float, optional
`-pre_route_clock_cap` *capFactor*	Specifies the scale factor for capacitance of clock nets in pre-route mode. *Default:* `0` When this parameter is not specified, the value specified with the `-post_route_cap` parameter is used for pre-route extraction. When specified, this parameter overrides the value specified with the `-post_route_cap` parameter for clock nets. *Data_type*: float, optional

`-pre_route_clock_res` *resFactor*	Specifies the scale factor for resistance of clock nets in pre-route mode. *Default:* `0` When this parameter is not specified, the value specified with the `-post_route_res` parameter is used for pre-route extraction. When specified, this parameter overrides the value specified with the `-post_route_res` parameter for clock nets. *Data_type*: float, optional
`-pre_route_res` *resFactor*	Specifies the resistance scale factor for RC extraction in pre-route mode. *Data_type*: float, optional
`-post_route_cap` `{`*capFactor1 capFactor2 capFactor3*`}`	Specifies the capacitance scale factor for RC extraction in post-route mode. *Default:* `1.0` *Data_type*: string, optional
`-post_route_clock_cap` `{`*capFactor1 capFactor2 capFactor3*`}`	Specifies the scale factor for capacitance of clock nets in the post-route extraction mode. *Default:* `0` When this parameter is not specified, the value specified with the `-post_route_cap` parameter is used for post-route extraction. When specified, this parameter overrides the value specified with the `-post_route_cap` parameter. *Data_type*: string, optional

`-post_route_clock_res {resFactor1 resFactor2 resFactor3}`	Specifies the scale factor for resistance of clock nets in the post-route extraction mode. *Default:* `0` When this parameter is not specified, the value specified with the `-post_route_res` parameter is used for postroute extraction. When specified, this parameter overrides the value specified with the `-post_route_res` parameter for clock nets. *Data_type*: string, optional
`-post_route_res {resFactor1 resFactor2 resFactor3}`	Specifies the resistance scale factor for RC extraction in post-route mode. *Data_type*: string, optional
`-post_route_cross_cap {xcapFactor1 xcapFactor2 xcapFactor3}`	Specifies the coupling capacitance scale factor for RC extraction in post-route mode. *Default:* `1.0` *Data_type*: string, optional
`-name rcCornerName`	Specifies the name for the RC corner object being created. You can choose any name for an RC corner object, but it is recommended that you use a name that can be easily recognized as referring to a specific RC corner (for example, rc-min, rc-max, c-min, c-max, and so on). *Data_type*: string, required

`-qrc_tech` *`fileName`*	Specifies the name of the Quantus technology (`.tch`) file used by TQuantus/IQuantus or Quantus sign-off RC extraction. The technology file is generated by Techgen, a built-in functionality of Quantus. For most technologies you can get this file from technology vendors. For technologies that are not available through the technology vendors, you enter the fabrication process information into an ASCII-format interconnect technology (ICT) input file. The ICT file is then used as input to IceCaps, which in turn generates the technology file. For more information on creating the ICT files, see the chapter, "Creating the ICT File", in the *Innovus Stylus Common UI User Guide*. *Data_type*: string, optional
`-temperature` *`rcTemperatureValue`*	Specifies the temperature, in units of Celsius, to use to derate resistance values. Use this parameter when you want to override the default temperature specified in the capacitance table or the Quantus technology file. **Note:** The software does not require that the temperature specified by the PVT operating condition at the delay calculation corner level be the same as the RC nominal or user-specified temperature value. *Default:* The Innovus system extraction system uses the default temperature of 25 degrees Celsius, unless it is overridden by the capacitance table or the Quantus technology file. *Data_type*: float, optional

`-via_variation_file` *viaVariationFile*	Specifies a file for modeling via variation. The VIA variation file lists up all the VIA layer names with the corresponding multiplier. Each RC corner can have its own VIA variation file.
	For example:
	`create_rc_corner`
	`-rc_corner rcCornerName1 \`
	`-via_variation_file viaVariationFileName1`
	`create_rc_corner`
	`-rc_corner rcCornerName2 \`
	`-via_variation_file viaVariationFileName2`

Examples

- The following command creates an RC corner called `rc-cbest` that uses the capacitance table `myTech_bc.CapTbl`, and derates the resistance values based on the temperature of 50 Celsius:
  `create_rc_corner -name rc-cbest -cap_table myTech.4_bc.CapTbl -temperature 50`

Related Information

- `update_rc_corner`
- "Design Import and Export in Stylus" chapter in the *Innovus Stylus Common UI User Guide*
  - Configuring the Setup for Multi-Mode Multi-Corner Analysis
  - Creating RC Corner Objects

create_timing_condition

```
create_timing_condition
[-help]
-library_sets string
-name timingConditionName
[-opcond string]
[-opcond_library string]
```

Describes a specific timing condition.

A timing condition is a set of libraries at a specific operating condition. Timing conditions are assigned to power domains or supply sets, and effectively describe the delay calculation requirements for the domains.

Parameters

`-help`	Prints out the command usage. For a detailed description of the command and all of its parameters, use the `man` command: `man create_timing_condition`.
`-library_sets string`	Specifies list of library sets to associate with this timing condition.
`-name timingConditionName`	Specifies a unique name for the timing condition.
`-opcond string`	Specifies a single name for an operating condition to associate with this timing condition.
`-opcond_library string`	Specifies name of library to search for the named operating condition. **Notes**: • The opcond library must be a library name, and not a library file path. • In a trilib flow, where multiple library sets of different PVTs are used for opcond interpolation, the opcond library should be close to the operating condition.

Related Commands

- update_timing_condition

delete_analysis_view

```
delete_analysis_view
[-help]
analysis_view_list
```

Deletes the specified analysis views.

Parameters

`-help`	Prints out the command usage.
`analysis_view_list`	Specifies the list of views for deleting.

Related Commands

- create_analysis_view
- update_analysis_view
- set_analysis_view
- report_analysis_views

get_capacitance_unit

`get_capacitance_unit`

Returns the session capacitance unit in picofarads.

Example

The following command returns the session capacitance unit:

get_capacitance_unit

1.000000pf

Related Information

- `set_library_unit`

get_clock_network_objects

```
get_clock_network_objects
[-help]
[-clocks clock_list]
[-type {cell | pin}]
```

Returns the cells or pins that exist in a clock network of the specified clocks.

Parameters

`-help`	Prints out the command usage.
`-clocks clock_list`	Returns the cells or pins present in a clock network of the specified clock(s).
`-type {cell | pin}`	Returns the specified cells or pins in the clock network.

Examples

- The following command returns the pins present in the clock network of all the clocks:
  ```
  get_clock_network_objects
  ```
- The following command returns the pins that exist in the clock network of CLK and CLK2 clocks:
  ```
  get_clock_network_objects -clocks {CLK CLK2}
  ```
- The following command returns the cells that exist in the clock network of CLK clock.
  ```
  get_clock_network_objects -clocks {CLK} -type cell
  ```

get_constant

```
get_constant
[-input_side | -output_side]
pin_port_obj
```

Queries the design database for the state of the specified pin or the state propagated through the combinational logic cone to that pin. The returned value is `0` for logic 0, `1` for logic 1, `X` for undefined, and `Z` for tristate.

Note: Constants on hierarchical ports are not computed.

The `get_constant` command returns the state for one pin only. If you want to know the state on all ports that have a constant set, use the `-type constant` option with the report_ports command. For example:
`report_ports -type constant`. Use the `-tcl_list` option to process the list in Tcl.

See report_nets examples for an example of the `-tcl_list` output.

Parameters

`-input_side | -output_side`	Returns the constant that is set on the input or output part of bidirectional pins. *Default*: Returns the constant on the output part.
pin_port_obj	Specifies the name of the single pin to query for pin state. The pin must be an instance pin or primary I/O port.

Examples

- The following command sets the value of the specified pin(s) to either a 1 or 0 for use by the timing engine: `set_case_analysis 0 J_block/zbuf0/A`
- The following command queries the design database for the state of the specified pin or the state

propagated through the combinational logic cone to that pin: `get_constant J_block/zbuf0/A`

Related Information

- `report_ports`
- `set_case_analysis`
- `report_case_analysis`

get_equivalent_cells

```
get_equivalent_cells
[-help]
-cell <string>
[[[{{-delay_corner <string> [-early_only | -late_only]} | -max | -min } [-power_domain
string]] | -library string]]
```

Returns equivalent cells from minimum/maximum libraries or from libraries of specific power domains.

Parameters

`-cell` *string*	Specifies the cell name for which equivalent cells will be fetched.
`-delay_corner` *string*	Specifies delay corner for which equivalent cells from libraries will be fetched.
`-early_only`	Fetches cells only for early libraries of the specified delay corner.
`-late_only`	Fetches cells only for late libraries of the specified delay corner.
`-library` *string*	Specifies the library for which cells will be fetched.
`-max`	Fetches equivalent cells for max libraries only.
`-min`	Fetches equivalent cells for min libraries only.
`-power_domain` *string*	Fetches cells for libraries of a specific power domain.

get_interactive_constraint_modes

`get_interactive_constraint_modes`

Returns a list of the multi-mode multi-corner constraint modes that are in interactive constraint entry mode.

When the software is in interactive constraint entry mode (`set_interactive_constraint_modes`), any timing constraints you specify take effect immediately on all active analysis views that are associated with these constraint modes.

Note: Interactive constraint mode only works when the software is in multi-mode multi-corner timing analysis mode.

Related Commands

- `set_interactive_constraint_modes`

get_propagated_clock

```
get_propagated_clock
[-clock clock_list] [-pin pin_list]
[> filename]
```

Returns the clock propagation mode, either `ideal` or `propagated`, for the specified clocks or pins.

Parameters

`> filename`	Specifies the name of a file in which to save the report. You can add the `.gz` extension to the file name to generate a compressed report. **Note:** The file name must be the last argument in the list.
`-clock clock_list`	Specifies one or more clock waveforms for which you want to retrieve the clock propagation mode information.
`-pin pin_list`	Specifies one or more pins or ports for which you want to retrieve the clock propagation mode information. You cannot specify hierarchical pins.

Example

- The following command gets the clock propagation mode information for all clock waveforms:
  get_propagated_clock -clock {*}

Related Information

- `set_propagated_clock`

get_resistance_unit

```
get_resistance_unit
```

Returns the resistance unit for the current session.

Example

The following command returns the resistance unit for the current session:

```
get_resistance_unit
```

1.000000 kohm

get_socv_reporting_nsigma_multiplier

```
get_socv_reporting_nsigma_multiplier
[-help]
-view viewName
{[-setup | -hold ]}
```

Allows you to query sigma multiplier setting for the specified view in setup or hold mode.

Parameters

`-help`	Prints out the command usage.
`-hold`	Returns the sigma multiplier for hold mode.
`-setup`	Returns the sigma multiplier for setup mode.
`-view viewName`	Returns the sigma multiplier value for the specified view. **Note**: Applies to active views only.

Examples

- The following command displays the sigma multiplier setting for the specified view:

```
innovus > set_socv_reporting_nsigma_multiplier -hold 2.5 -view
default_emulate_view_hold
innovus > get_socv_reporting_nsigma_multiplier -view default_emulate_view_hold -hold
2.500000
```

Related Commands

- set_socv_reporting_nsigma_multiplier

get_time_unit

`get_time_unit`

Returns the current time unit for the session.

Example

- The following command returns the time unit for the session:
  get_time_unit

  1.000000ns

Related Information

- `set_library_unit`

gui_show_timing_map

```
gui_show_timing_map
[-help]
{[-step <float> -threshold <float> -check_type {setup hold}] | -clear }
```

Displays the timing map in GUI.

Parameters

`-help`	Prints out the command usage
`-check_type {setup | hold}`	Specifies the check_type.
`-clear`	Clear the timing map.
`-step <float>`	Specifies the step.
`-threshold <float>`	Specifies the slack threshold.

merge_timing_reports

```
merge_timing_reports
[-help]
[-base_dir string]
-base_report string
[-command {report_constraint | report_timing | report_clock_timing}]
[-max_paths integer]
[-nworst integer]
[-text]
-view_dirs view_list
> <file_name>
```

Merges machine-readable/text timing reports from the specified analysis views into a single report file. The software merges the reports based on the slack value. The path with the most critical slack is listed first in the report.

The `merge_timing_reports` command can be used for both single CPU and distributed multi-mode multi-corner timing analysis.

Parameters

`-help`	Prints out the command usage.
`-base_dir` *`string`*	Specifies the path to the output directory where the view-specific directories exist. The software looks in this directory for the base file name in the view specific directories, and then places the merged output in the specified redirection file. If it is not specified then software uses the default base directory.
`-base_report` *`string`*	Specifies the base file name of the timing reports to merge
`-command {report_constraint | report_timing | report_clock_timing}`	Merges the text timing reports from the specified file names. The generated timing reports can be from the commands `report_constraint`, `report_timing`, or `report_clock_timing`.
`-max_paths` *`integer`*	
`-nworst` *`integer`*	Specifies to have at the most n paths per end point in a merged report.
`-text`	Generates merged report in the text format. **Note**: You can use this option only when the input reports are machine-readable.
`-view_dirs` *`view_list`*	Merges timing reports from the specified directories.
`>` *`file_name`*	Specifies the name of the file in which to merge the timing reports.

Examples

Here are some examples.

- The following command merges the machine-readable timing reports with the base file name `rt_setup1.rpt` in the analysis views `v1`, `v2`, and `v3` into a single machine readable timing report called `merged_report.rpt`:

  ```
  merge_timing_reports -view_dirs {v1 v2 v3} -base_report rt_setup1.rpt >
  merged_report.rpt
  ```

- The following command merges the machine-readable timing reports with the base file

name rt_setup1.rpt in the analysis views v1, v2, and v3 into a single text timing report called `merged_report.rpt`:

```
merge_timing_reports -view_dirs {v1 v2 v3} -base_report rt_setup1.rpt -text >
merged_report.rpt
```

- The following commands merge text timing reports (from `report_timing` command) with the base file name `topPaths.rpt` in the analysis views `v1` and `v2` into a single timing report called `merged_topPaths.rpt`:

```
distribute_command {report_timing -max_paths 10 > topPaths.rpt}
merge_timing_reports -command report_timing -view_dirs {v1 v2} -base_report
topPaths.rpt -base_dir dmmmc_run_dir > merged_topPaths.rpt
```

- The following commands merge text timing reports (from `report_constraint` command) with the base file name `allViol.rpt` in the analysis views `v1` and `v2` into a single timing report called `merged_allViol.rpt`:

```
distribute_command {report_constraint -all_violators > allViol.rpt}
merge_timing_reports -command report_constraint -view_dirs {v1 v2} -base_report
allViol.rpt -base_dir dmmmc_run_dir > merged_allViol.rpt
```

Related Commands

- report_timing
- report_constraint

read_path_descriptions

```
read_path_descriptions
[-help]
filename
[-hpin]
[-path_type {end summary full full_clock end_slack_only summary_slack_only}]
[-retime {aocv ssta path_slew_propagation aocv_path_slew_propagation
waveform_propagation}]
```

Reads the file created by the `write_path_descriptions` command. This command reads the complete description of the paths specified in the file in another session.

The paths description when read back can be saved as a collection that can be used as a regular path collection for any reporting command or queried using the `get_property` command.

Parameters

`-help`	Prints out the command usage.
`filename`	Specifies the name of the input file.
`-hpin`	Prints all the hierarchical crossings of an arc under the "hpin" column in the `report_timing` table.
`-path_type {end | summary | full | full_clock | end_slack_only | summary_slack_only}`	Allows you to choose the format of the report based on the path type.
`-retime {aocv | ssta | path_slew_propagation | aocv_path_slew_propagation | waveform_propagation}`	Reanalyzes the specified set of paths using the specified analysis method.

Examples

- The following example command reads out the pathfile1.txt. The file contains details of the collection of timing paths with hierarchical pins (`-hpin`) format based on timing and exact path availability in the session in which the path description is read:

  ```
  read_path_descriptions pathfile1.txt -hpin
  ```

Related Commands

- `write_path_descriptions`

read_sdf

```
read_sdf
[-help]
sdf_file_name
[-continue_on_error]
[-ilm_filter]
[-incremental]
[-overwrite_incremental_delay]
[-path instanceName]
[-persistent]
[-report_summary]
[-scale float]
[-strict_cond_matching]
[-view viewName]
[-sdf_field {min|typ|max} | [-early_sdf_field {min|typ|max}] [-late_sdf_field
{min|typ|max}]]
```

Reads an OVI Standard Delay Format (SDF) file and annotates user-specified delay information from the SDF file into the timing system. You can use the `read_sdf` command to supply precalculated delays and timing check values from an external delay calculator, or from a previous Innovus System session.

The `read_sdf` command can read version 2.1 and 3.0 of the SDF files, and can read both plain-text and GNU zipped (`.gz`) format files. The command and the Innovus System software do not support the Timing Environment section of the SDF specification.

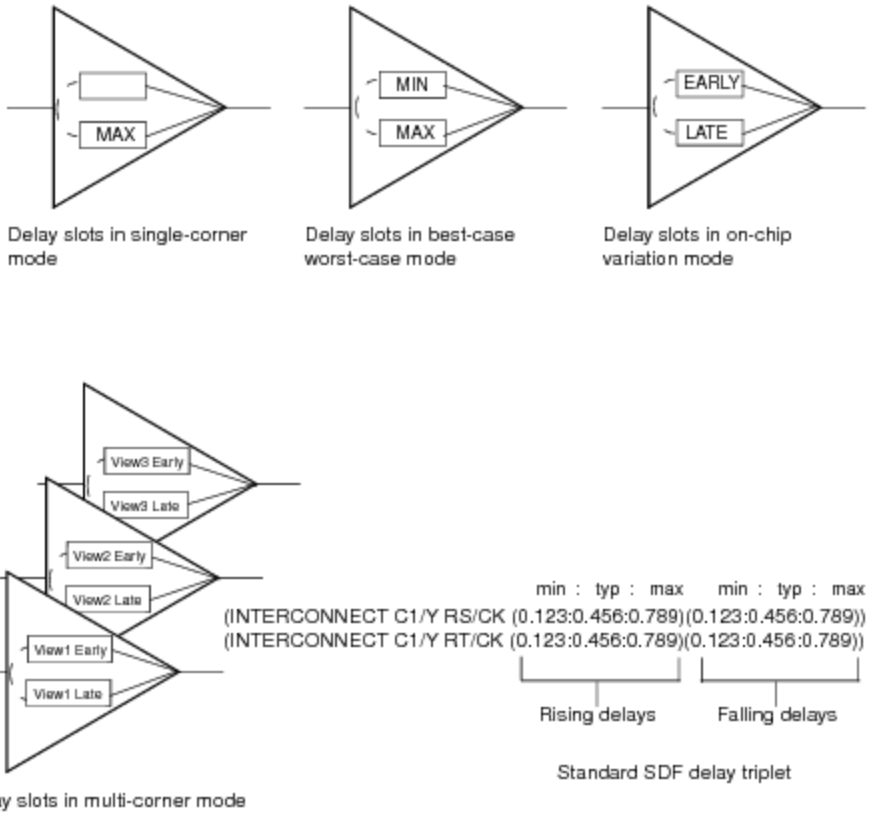
The SDF file can provide one, two, or three different values for each timing arc delay or timing check value. These values are passed in the min, typ, or max SDF delay fields, represented as min:typ:max.

The Innovus System software provides multiple delay place holders to store the calculated or annotated delay information:

- In single-corner analysis mode, only max delays are available.
- In best-case worst-case mode, a min delay slot is provided for storing min delays used for hold analysis. A max delay slot stores delays used for setup analysis.
- In on-chip variation mode, the early delay slots store delays used for calculated early arrival times. The late delay slot stores delays used for calculated late arrival times.
- In multi-corner mode, an early and a late delay slot are provided for each multi-mode multi-corner analysis view.

The `read_sdf` command provides parameters for controlling how the SDF min:typ:max triplet values

are mapped into the Innovus System software's delay slots. The correct mapping depends on how the delay information in the SDF file was created, and what type of analysis you are performing.



The timing system provides special handling for annotating incremental delays. These delays typically represent plus or minus offsets due to crosstalk effects. Incremental delay annotation to the Innovus System software can be done by:

- Importing an incremental SDF file (SDF has an internal `INCREMENT` keyword) using the `read_sdf` command.
- Tcl sourcing a file of `set_annotated_delay -net -delta_only` constraints.

The incremental delay annotations can be viewed by adding the `incr_delay` option to the `report_timing -fields` list, or to the `timing_report_fields` attribute.

Parameters

`-help`	Prints out the command usage.

`-continue_on_error`	Continues to read the file if an error occurs. By default, when an error occurs, the `read_sdf` command stops reading the file. Use this parameter to attempt to continue reading the file, if possible. **Note:** Use caution if you continue, because some annotations could be missing due to the errors. Use the `report_annotated_parasitics` command to find missing annotations. If the file is corrupted, obtain a new SDF file.
`-early_sdf_field {min | typ | max}`	Annotates the early delay from the specified SDF field. In single-corner mode, all setup and hold analysis uses max delays. The software ignores any min/early delay information. In best-case worst-case mode, use this parameter to specify which SDF field to use for hold analysis. In on-chip variation mode, the software uses the early delay for calculation of fast paths.
`-ilm_filter`	Ignores ILM-related errors. By default, the `read_sdf` command does not ignore errors reported due to missing arcs in core ILM.
`-incremental`	Increments the total delay without affecting the delta delay. By default, when the SDF file contains the INCREMENT delay type, the `read_sdf` command will perform an adjustment on the delta delay slot which models SI cross-talk push-out/pull-in effects. When this option is used, the INCREMENT delays will be applied against the static base delay values instead.

`-late_sdf_field {min | typ | max}`	Annotates the late delay from the specified SDF field. In single-corner mode, use this parameter to specify which of the SDF triplet values to annotate for both setup and hold checks. In best-case worst-case mode, use this parameter to specify which SDF field to use for setup analysis. In on-chip variation mode, the software uses the late delay for the calculation of slow paths.
`-overwrite_incremental_delay`	When annotating an incremental SDF file, replaces any pre-existing delta delays with the new SDF values. The total delay will be equivalent to the original base delay, plus the new delta delay. If the delays in the incremental SDF file represent crosstalk delay push-outs from a signal integrity analysis, use the `-overwrite_incremental_delay` parameter to ensure that any stale data from a previous analysis is removed. By default, when an incremental SDF file (SDF has an internal `INCREMENT` keyword) is annotated, the delays are added to the base delay, and also stored separately as delta delays. If an additional incremental SDF file is annotated, the software adds the existing base and delta delays with the new values.
`-path` *instanceName*	Specifies the instance in the design to which the specified SDF file should be applied.
`-persistent`	Keeps the delays from certain SDF files persistently in use during various steps of the timing optimization . Use this parameter to support pre-timed structures, such as clock meshes.
`-report_summary`	Outputs the summary information of annotated delays.

`-scale` *float*	Overrides the automatic scaling of SDF delays to the library units by the value given. To disable all scaling, use a value of 1.0.
`-sdf_field {min | typ | max}`	Annotates the delay from the specified SDF slot. In single-corner and best-case worst-case modes, the software uses this value for both setup and hold analysis. In on-chip variation mode, the software uses the same value from the specified SDF slot for both the early and late delays of the specified arc.
sdf_file_name	Specifies the name of the SDF file that has the back annotation data.
`-strict_cond_matching`	Annotates only the conditions that are an exact syntax-to-syntax match between the SDF file and the `.lib` file. *Default:* Uses looser conditional matching rules between the `.lib` and SDF arc specifications.
`-view` *viewName*	Annotates the specified analysis view(s) with the delays from the SDF file. You can use this parameter only when you are in multi-mode multi-corner analysis mode.

Examples

- The following command annotates delay information from the SDF min field in to the min/early slots, and maps the SDF max field data to the max/late delay slots:

  `read_sdf` *sdf_file_name*

  Given the following SDF, the default annotation would be:

  (INTERCONNECT C1/Y RS/CK (0.123:0.456:0.789) (0.123:0.456:0.789))

  - In single-corner analysis mode, a delay of 0.789 ns is annotated to the max delay slot.
  - In best-case worst-case mode, a delay of 0.123 ns is annotated to the min delay slot, and a value of 0.789 ns is placed in the max delay slot.

    - In on-chip variation mode, a delay of 0.123 is used for early delays, and a value of 0.789 ns is used for late delays.
- The following command annotates the SDF typ field to both the early and late delay slots:
  `read_sdf -sdf_field typ sdf_file_name`
- The following command annotates the SDF typ field to the early delay slot, and the SDF max field to the late delay slot:
  `read_sdf -early_sdf_field typ -late_sdf_field max sdf_file_name`
- The following command annotates the SDF typ field to the late delay slot. The early delay slot is annotated with the default min SDF field:
  `read_sdf -late_sdf_field typ sdf_file_name`

Related Commands

- write_sdf
- set_annotated_delay
- report_timing
- report_annotated_parasitics

read_transition_file

```
read_transition_file
-file fileName
[-view viewName]
```

Reads external transition files that are used to perform maximum transition violation reporting and fixing.

The contents of a transition file must be in the following format:

```
setTranTime [-add] [-maxR slewValue | -maxF slewValue] {pinName}
```

or

```
setTranTime [-add] [-minR slewValue | -minF slewValue] {pinName}
```

Where:

- `-add`: Specifies that the transition value is incremental.
- `-maxR | -maxF`: Specifies the maximum rise or fall transition.
- `-minR | -minF`: Specifies the minimum rise or fall transition.

  Note: The minimum transition values will be ignored while performing transition violation fixing.
- *`pinName`*: Specifies the pin name for which the transition value is specified. To specify multiple pin names, you need to use separate `setTranTime` statements as given below:

  ```
  setTranTime -add -maxR 3.5 inst/A
  setTranTime -add -maxR 3.5 pinB
  ```

Note: The `read_transition_file` command uses the time unit that is written in the slew file. If time unit is specified as 1e-12, it reads the values in picoseconds. For time unit specified as 1e-09, the command reads the values in nanoseconds. For example,

```
# Time Unit : 1e-09 seconds

setTranTime -minR 0.80050 {AAA/BBB/CCC/A}

setTranTime -minF 0.580300 {AAA/BBB/DDD/A}
```

When a transition file is read and third-party slew values are annotated, Innovus uses the following command to report transition violations with the annotated slews:

```
time_design {-postRoute | signoff | reportOnly -si} -useTransitionFiles
```

Parameters

`-file` *`fileName`*	Specifies the name of the file that contains information about the transition time.
`-view` *`viewName`*	Specifies the view name for which the transition file will be read in case of MMMC designs. You need to ensure that you specify the transition files for all setup views to perform transition violation fixing. **Note**: This parameter is required for MMMC designs only.

Example

- The following command reads an external transition file called `sampleFile` for non-MMMC designs:

  ```
  read_transition_file -file sampleFile
  ```

- The following commands read view specific external transition files for a MMMC design:

  ```
  read_transition_file -view view1 -file sampleFile1
  read_transition_file -view view2 -file sampleFile2
  ```

read_twf

```
read_twf
[-help]
[filenames]
[-cell string]
[-prefix string]
[-quiet]
[-reset]
[-scope string]
[-skip_constant]
[-skip_timing_window]
[-strip_prefix string]
[-verbose]
[-view string]
```

Reads the external timing window file (TWF) for noise (Signal Integrity) and power calculation (Common Power Engine - CPE). This file is used to obtain clocks, constants, external load, slews, timing windows, and slack information. The external TWF specification is optional. In the absence of a TWF, this information is automatically derived using the Common Timing Engine (CTE), if timing constraints are read.

Note: If the `VOLTAGE_THRESHOLD` construct is missing from the TWF file header, the values 10 and 90 are automatically assigned as the minimum and maximum threshold values, respectively. For example:

```
VOLTAGE_THRESHOLD (10 90)
```

Parameters

`-help`	Prints out the command usage.
`-cell` *string*	Specifies the master cell for the timing window file.
filenames	Specifies the names of the timing window files.
`-prefix` *string*	Adds a prefix to the net or pin name.
`-quiet`	Skips information and warning messages.
`-reset`	Clears the previously read TWF files.
`-scope` *string*	Specifies the scope for the given hierarchical timing window file.
`-skip_constant`	Skips constant processing.
`-skip_timing_window`	Skips timing windows processing.
`-strip_prefix` *string*	Removes hierarchical path name.
`-verbose`	Prints the command progress details.
`-view` *string*	Specifies the view for the timing window file.

Examples

- The following command reads the timing window file TW.in for the view 'view01', while skipping the constant processing from the timing window file:
  ```
  read_twf TW.in -skipconst -view view01
  ```
- The following command clears the previously read TWF file:
  ```
  read_twf -reset
  ```

Related Commands

- `write_timing_windows`

report_analysis_coverage

```
report_analysis_coverage
[-help]
[pin | port]
[-check_type check_type_list]
[-exclude_untested reason_list]
[-include_dynamic_checks]
[-include_parallel_check_arcs]
[-include_when_cond]
[-max_paths{type integer}]
[-nworst <integer>]
[-retime {aocv | ssta | path_slew_propagation | aocv_path_slew_propagation}]
[-sort {pin | refpin | checktype | slack | reason}]
[-tcl_list]
[-unsorted]
[-verbose check_status_list]
[-view view_name]
[> <filename[.gz]>]
[>> <filename[.gz]>]
```

Provides information about the timing checks in the design. Valid check types are clock_gating_hold, clock_gating_pulse_width, clock_gating_setup, clock_period, clock_separation, external_delay, hold, no_change_hold, no_change_setup, path_delay, pulse_width, recovery, removal, setup, skew, and time_borrow. Valid status types are: met, untested, and violated.

For each type of timing check, the command reports the number of checks that meet constraints, violated constraints, or that are untested in the `SUMMARY` section. The `DETAILS` section includes information about the signal pin, reference pin, check type, slack, and the reason for being untested.

A `met` check shows as a positive value in the `Slack` column. A `violated` check shows as a negative value in the `Slack` column. An `untested` check shows as `UNTESTED` in the `Slack` column and the reason is given in the `Reason` column. The `Reason` column is blank when a `Slack` number is given because the check has been tested.

Note: If there are no timing checks in the design of the type specified by `-check_type`, the report shows "No timing checks found."

Use `check_timing` to look for missing constraints.

Parameters

`-help`	Prints out the command usage.
`{> | >>} file_name`	When specified with `>`, writes the report to the specified file name. If the file already exists, the software overwrites it. When specified with `>>`, writes the report to the specified file name. If the file already exists, the software concatenates the report to the end of the file. The *`filename`* parameter must be the last argument in the list. The *`filename`* and `-tcl_list` parameters are mutually exclusive; you cannot specify them together. **Note:** To write a compressed report, add the `.gz` extension to the file name. *Default:* Report is displayed on standard output without being saved.

`-check_type` *check_type_list*	Limits the report to the type of check(s) in the *check_type_list*. *Default*: All (all types of check(s) are reported) Valid check types are: • `setup`: Reports paths with setup timing checks. • `hold`: Reports paths with hold timing checks. • `pulse_width`: Reports paths with pulse width timing checks. • `clock_period`: Reports paths with clock period timing checks. • `clock_gating_setup`: Reports paths with clock gating setup timing checks. • `clock_gating_hold`: Reports paths with clock gating hold timing checks. • `clock_gating_pulse_width`: Reports paths with clock gating pulse width checks. • `recovery`: Reports paths with recovery timing checks. • `removal`: Reports paths with removal timing checks. • `clock_separation`: Reports paths with clock separation timing checks. • `skew`: Reports paths with skew checks. • `data_setup`: Reports paths with data to data setup checks. • `data_hold`: Reports paths with data to data hold checks.

`-exclude_untested {`*`reason_list`*`}`	Excludes from the report endpoints that are unconstrained due to a specific reason. Valid reasons are: • `const`: Constant reaching signal/reference pin of any timing check. • `false_path`: Path disabled due to `set_false_path`. • `no_endpoint_clock`: Missing clock at the reference pin of the timing check. • `no_startpoint_clock`: Missing clock at the signal pin of the timing check. This may happen if the clock signal does not reach the launching register. • `unknown`: None of the known reasons are identified for an untested path. • `user_disable`: Checks arc disable due to `set_disable_timing`.
`-include_dynamic_checks`	Reports the dynamic checks (data_to_data, path_delay) on pins that are constrained by set_data_check, set_max_delay, or set_min_delay constraints. By default, these checks will not be reported and the command only reports the pins constrained by all library checks, set_output_delay or inferred clock gating checks. The dynamic checks, such as set_data_check or set_max_delay are not reported by default, when `-include_dynamic_checks` parameter is not specified.
`-include_parallel_check_arcs`	Reports all the parallel check arcs that have been merged.
`-include_when_cond`	Reports the when condition of check arcs.
`-max_paths{type` *`integer`*`}`	Enumerates the specified number of worst paths in the design, regardless of the endpoint.

`-nworst` *integer*	Enumerates the specified number of paths for each endpoint.
`-retime {aocv | ssta | path_slew_propagation | aocv_path_slew_propagation}`	Reanalyzes the set of paths using the specified method.
pin | *port*	Limits the report to timing checks on specified pins or ports.

`-sort {pin | refpin | checktype | slack | reason}`	Sorts the checks by the specified method while reporting detailed information using the `-verbose` option. The `-sort` option does not change the order of the columns. The criteria by which sorting is attempted for the `slack`, `pin`, and `checktype` arguments is as follows: • When sorting by slack, if the slack values are the same, the software then attempts to sort the checks by pin name. If the pin names are the same, the software attempts to sort by timing check type. When you sort by `slack`, the slack value is reported in order of least negative to most negative (worst slack last). • When sorting by pin name, if the pin names are the same, the software then attempts to sort the checks by slack value. If the slack values are the same, the software attempts to sort by timing check type. • When sorting by timing check type, if the check types are the same, the software then attempts to sort by slack value. If the slack values are the same, the software attempts to sort by pin name. *Default*: By default, checks are sorted by slack value (`slack`). Improve usability of the report for debugging purposes by using the -verbose and -sort options as shown in these examples: `report_analysis_coverage -verbose untested -sort reason report_analysis_coverage -verbose violated -sort slack`

`-tcl_list`	Produces the report in Tcl list format instead of a tabular format. This is useful for integrating timing with custom Tcl functions, and also for customizing report generation. The `-tcl_list` and *`filename`* parameters are mutually exclusive; you cannot specify them together.
`-unsorted`	Reports unsorted slacks with untested end points at the end of the report.
`-verbose` *`check_status_list`*	Limits the detailed information to only the checks with the status specified. Valid status types are: `met`, `violated`, and `untested`. Reduce the number of lines in the `DETAILS` section and improve usability of the report by using the `-verbose` option to filter out the `met` checks as shown in this example: `report_analysis_coverage -verbose {violated untested}` **Note**: The `-verbose` parameter supports multi-threading.
`-view` *`view_name`*	Generates a timing check report for the specified analysis view only. You can specify this parameter only when the software is in multi-mode multi-corner analysis mode.

Examples

- The following command generates the following report about the timing checks in the design:
  `report_analysis_coverage`

```
   +-------------------------------------------------------------+
   |                TIMING CHECK COVERAGE SUMMARY                |
   |-------------------------------------------------------------|
   |      Check Type       | No. of |   Met    | Violated | Untested |
   |                       | Checks |          |          |          |
   |-----------------------+--------+----------+----------+----------|
   | Clock Gating Hold     | 2      | 0 (0%)   | 2 (100%) | 0 (0%)   |
   | Clock Gating Setup    | 2      | 2 (100%) | 0 (0%)   | 0 (0%)   |
   | ExternalDelay (Early) | 4      | 4 (100%) | 0 (0%)   | 0 (0%)   |
   | ExternalDelay (Late)  | 4      | 4 (100%) | 0 (0%)   | 0 (0%)   |
   | Hold                  | 16     | 4 (25%)  | 2 (12%)  | 10 (62%) |
   | PulseWidth            | 16     | 14 (87%) | 0 (0%)   | 2 (12%)  |
   | Setup                 | 16     | 6 (37%)  | 0 (0%)   | 10 (62%) |
   +-------------------------------------------------------------+
+-------------------------------------------------------------------------------+
|                          TIMING CHECK COVERAGE DETAILS                        |
|-------------------------------------------------------------------------------|
|     Pin      |  Reference   |      Check Type       |  Slack   |   Reason     |
|              |     Pin      |                       |          |              |
|--------------+--------------+-----------------------+----------+--------------|
| BLK/BR1/CK ^ | BLK/BR1/CK v | PulseWidth            | UNTESTED | const        |
| BLK/BR1/CK v | BLK/BR1/CK ^ | PulseWidth            | UNTESTED | const        |
| BLK/BR1/D    | BLK/BR1/CK ^ | Hold                  | UNTESTED | const        |
| BLK/BR1/D    | BLK/BR1/CK ^ | Setup                 | UNTESTED | const        |
| BLK/BR2/CK ^ | BLK/BR2/CK v | PulseWidth            |    4.854 |              |
| BLK/BR2/CK v | BLK/BR2/CK ^ | PulseWidth            |    3.972 |              |
| BLK/BR2/D    | BLK/BR2/CK ^ | Setup                 | UNTESTED | disable      |
| BLK/BR2/D    | BLK/BR2/CK ^ | Hold                  | UNTESTED | disable      |
| CG1/A ^      | CG1/B v      | Clock Gating Hold     |   -5.041 |              |
| CG1/A ^      | CG1/B ^      | Clock Gating Setup    |    9.943 |              |
| CG1/A v      | CG1/B v      | Clock Gating Hold     |   -5.072 |              |
| CG1/A v      | CG1/B ^      | Clock Gating Setup    |    9.979 |              |
| CGR/CK ^     | CGR/CK v     | PulseWidth            |    4.610 |              |
| CGR/CK v     | CGR/CK ^     | PulseWidth            |    4.360 |              |
| CGR/D        | CGR/CK ^     | Hold                  | UNTESTED | No data signal |
| CGR/D        | CGR/CK ^     | Setup                 | UNTESTED | No data signal |
|                                                                               |
```

- The following command filters from the report all endpoints that are unconstrained because of the reason `const`:

  ```
  report_analysis_coverage -exclude_untested const
  ```

```
+-------------------------------------------------------------+
|                TIMING CHECK COVERAGE SUMMARY                |
|-------------------------------------------------------------|
|      Check Type       | No. of |   Met    | Violated | Untested |
|                       | Checks |          |          |(excluding|
|                       |        |          |          | ignored  |
|                       |        |          |          | reasons) |
|-----------------------+--------+----------+----------+----------|
| Clock Gating Hold     | 2      | 0 (0%)   | 2 (100%) | 0 (0%)   |
| Clock Gating Setup    | 2      | 2 (100%) | 0 (0%)   | 0 (0%)   |
| ExternalDelay (Early) | 4      | 4 (100%) | 0 (0%)   | 0 (0%)   |
| ExternalDelay (Late)  | 4      | 4 (100%) | 0 (0%)   | 0 (0%)   |
| Hold                  | 16     | 4 (25%)  | 2 (12%)  | 10 (62%) |
| PulseWidth            | 16     | 14 (87%) | 0 (0%)   | 2 (12%)  |
| Setup                 | 16     | 6 (37%)  | 0 (0%)   | 10 (62%) |
+-------------------------------------------------------------+
```

```
+-------------------------------------------------------------------------------+
|                          TIMING CHECK COVERAGE DETAILS                        |
|-------------------------------------------------------------------------------|
|     Pin      |  Reference   |      Check Type       |  Slack   |    Reason     |
|              |     Pin      |                       |          |               |
|--------------+--------------+-----------------------+----------+---------------|
| BLK/BR2/CK ^ | BLK/BR2/CK v | PulseWidth            |    4.854 |               |
| BLK/BR2/CK v | BLK/BR2/CK ^ | PulseWidth            |    3.972 |               |
| BLK/BR2/D    | BLK/BR2/CK ^ | Hold                  | UNTESTED | disable       |
| BLK/BR2/D    | BLK/BR2/CK ^ | Setup                 | UNTESTED | disable       |
| CG1/A ^      | CG1/B v      | Clock Gating Hold     |   -5.041 |               |
| CG1/A ^      | CG1/B ^      | Clock Gating Setup    |    9.943 |               |
| CG1/A v      | CG1/B v      | Clock Gating Hold     |   -5.072 |               |
| CG1/A v      | CG1/B ^      | Clock Gating Setup    |    9.979 |               |
| CGR/CK ^     | CGR/CK v     | PulseWidth            |    4.610 |               |
| CGR/CK v     | CGR/CK ^     | PulseWidth            |    4.360 |               |
| CGR/D        | CGR/CK ^     | Hold                  | UNTESTED | No data signal |
| CGR/D        | CGR/CK ^     | Setup                 | UNTESTED | No data signal |
+-------------------------------------------------------------------------------+
```

- When the software is in multi-mode multi-corner timing analysis mode, You can specify the `-view` parameter to generate a timing check report for the specified analysis view only:

  ```
  report_analysis_coverage -view s1
  ```

```
+-------------------------------------------------------------+
|       TIMING CHECK COVERAGE SUMMARY (Analysis View:  s1)    |
|-------------------------------------------------------------|
|      Check Type       | No. of |   Met    | Violated | Untested |
|                       | Checks |          |          |          |
|-----------------------+--------+----------+----------+----------|
| ExternalDelay (Early) | 1      | 1 (100%) | 0 (0%)   | 0 (0%)   |
| ExternalDelay (Late)  | 1      | 0 (0%)   | 1 (100%) | 0 (0%)   |
| Hold                  | 2      | 1 (50%)  | 1 (50%)  | 0 (0%)   |
| PulseWidth            | 4      | 4 (100%) | 0 (0%)   | 0 (0%)   |
| Setup                 | 2      | 1 (50%)  | 1 (50%)  | 0 (0%)   |
+-------------------------------------------------------------+
```

Related Information

- check_timing
- report_timing

report_analysis_summary

```
report_analysis_summary
[-help]
[-csv string]
[-merged_views]
[-retime {path_slew_propagation | aocv_path_slew_propagation}]
[-retime_mode {exhaustive}]
[-view viewName]
[> | >> filename[.gz]]
[-early | -late ]
[-expanded_summary_clocks | -merged_groups]
```

Generates timing slack summary reports for specified analysis view(s). This command categorizes the slack summary in terms of groups - In2reg, Reg2reg, Reg2Out, and In2Out. A column "ALL" is displayed that regroups in2reg, in2out, reg2reg, and reg2out details.

By default, the slack is categorized in terms of worst negative slack (WNS), total negative slack (TNS), and the total number of violations in group-based mode.

The `report_analysis_summary` command is supported in path-based analysis (PBA) mode.

Parameters

`-help`	Prints out the command usage.
`-csv string`	Specifies the name of the CSV output file.
{> | >>} *filename*	When specified with >, writes the report to the specified file name. If the file already exists, the software overwrites it. When specified with >>, writes the report to the specified file name. If the file already exists, the software concatenates the report to the end of the file. **Note:** To write a compressed report, add the .gz extension to the file name. *Default:* Report is displayed on standard output without being saved.

`-early`	Generates report for hold analysis mode for early paths.
`-expanded_summary_clocks`	Generates report in terms of each possible clock domain pair separately.
`-late`	Generates report for setup analysis mode for late paths.
`-merged_groups`	Merges all the clock groups for a consolidated timing summary.
`-merged_views`	Generates a merged report across all the views.
`-retime {path_slew_propagation | aocv_path_slew_propagation}`	Reanalyzes the defined set of paths using the specified method. By default, the timing report shows a maximum of 2 million paths for retime PBA analysis.
`-retime_mode {exhaustive}`	Reanalyzes the specified set of paths using the specified retime mode.
`-view` *`viewName`*	Generates report for the specified analysis views only. By default, all the active views are analyzed.

Example

The `report_analysis_summary` command shows the following report output:

```
REPORT ANALYSIS SUMMARY (View:default_emulate_view) (EL:Setup)
================================================================================
GroupType:Standard
================================================================================

All IN2REG IN2OUT REG2REG REG2OUT
--------------------------------------------------------------------------------
WNS:TNS:#VIO WNS:TNS:#VIO WNS:TNS:#VIO WNS:TNS:#VIO WNS:TNS:#VIO FromClk->ToClk
--------------------------------------------------------------------------------
-3.765:-6.872:3 -3.765:-3.765:1 0.000:0.000:0 -1.004:-1.004:1 -2.103:-2.103:1 *->clk1
--------------------------------------------------------------------------------
-9.304:-9.304:1 0.000:0.000:0 0.000:0.000:0 -9.304:-9.304:1 0.000:0.000:0 *->clk2
--------------------------------------------------------------------------------
-4.962:-23.213:6 0.000:0.000:0 0.000:0.000:0 -4.962:-23.213:6 0.000:0.000:0 *->clk3
--------------------------------------------------------------------------------
```

Related Commands

- set_default_view
- set_analysis_view

report_analysis_views

```
report_analysis_views
-help
[-type {all | setup | hold | active}]
```

Generates a hierarchical report of the current multi-mode multi-corner configuration. By default, the `report_analysis_views` command generates a report of all defined views in the design. Use the `-type` parameter to generate a report on a specific type of analysis view.

Use this command after creating analysis views (`create_analysis_view`), and setting active setup and hold views (`set_analysis_view`) for the design.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `report_analysis_views` parameter. For a detailed description of the command and all of its parameters, use the man command: `man report_analysis_views`.
`-type {all | setup | hold | active}`	Specifies the type of report to generate. *Default:* `all` • active: Generates a report of all active setup and hold views in the design. • all: Generates a report of all defined views in the design, including those that are currently inactive. • hold: Generates a report of all active hold views in the design. • setup: Generates a report of all active setup views in the design.

Related Commands

- create_analysis_view
- set_analysis_view

report_annotated_assertions

```
report_annotated_assertions
-type {load | transition}
[-list_annotated]
[-list_non_annotated]
[-view viewName]
[-max_lines number]
[-min | -max | -min_max]
obj_list
[{> | >>} file_name]
```

Reports information for annotated assertions, set using the `set_load` or `set_annotated_transition` command, according to the specified assertion type. You can select load or transition reporting.

Note: The `report_annotated_assertions` command is now obsolete. This command will continue to be supported in the current release, but will be removed in the next major release of the software.

Parameters

`{> | >>} file_name`	Writes the report to the specified file name. When specified with >, writes the report to the specified file name. If the file already exists, the software overwrites it. When specified with >>, writes the report to the specified filename. If the file already exists, the software concatenates the report to the end of the file.
`-list_annotated`	Lists the detailed information for the specified assertion types.
`-list_non_annotated`	Lists information for non-annotated assertions.

`-min	-max	-min_max`	Specifies to report minimum, maximum, or both values of the specified assertion type. When input assertions are applied using both -min and -max parameters, you can specify the -min_max parameter to report both the assertion values. In this case, you do not need to specify this command twice, that is, separately with -min and -max parameters. *Default*: max
`-max_lines` *number*	Controls the number of output lines reported by `-list_annotated` or `-list_non_annotated` parameter. By default, the specified net, pin, or port names are reported.
obj_list	Specifies the list of nets, pins, or ports to be reported.
`-type {load	transition}`	Specifies the assertion type, load or transition, to be reported.
`-view` *viewName*	Reports the annotations for the specified analysis view. In multi-mode multi-corner timing analysis mode, annotations are reported for the specified mode or corner.

Examples

- The following command reports all the load annotated assertions:
  ```
  report_annotated_assertions -type load -list_annotated -min_max
  ```

Related Commands

- `set_load`
- `set_annotated_transition`

report_annotated_check

```
report_annotated_check
[-check_type check_type_list]
[-max_line number]
[instance_list]
[-list_annotated]
[-list_non_annotated]
[-view viewName]
[> | >> filename[.gz] | -tcl_list]
```

Reports coverage of annotated timing checks.

Parameters

`-check_type` *check_type_list*	Allows you to specify the check type. The valid check types are `setup`, `hold`, `pulse_width`, `clock_period`, `clock_gating_setup`, `clock_gating_hold`, `clock_gating_pulse_width`, `data_setup`, `data_hold`, `recovery`, `removal`, `clock_separation`, `no_change_setup`, `no_change_hold`, and `skew`.

`> filename`	When specified with `>`, writes the report to the specified file name. If the file already exists, the software overwrites it. When specified with `>>`, writes the report to the specified file name. If the file already exists, the software concatenates the report to the end of the file. The *`filename`* parameter must be the last argument in the list. The *`filename`* and `-tcl_list` parameters are mutually exclusive; you cannot specify them together. **Note:** To write a compressed report, add the `.gz` extension to the file name. *Default:* Report is displayed on standard output without being saved.
`instance_list`	Specifies instance names for which annotations of check arcs are to be reported.
`-list_annotated`	Lists the detailed annotation information for the following timing checks: • setup • hold • recovery • removal • pulse-width • nochange-setup • clock-period • skew **Note**: If the software is in multi-mode multi-corner timing analysis mode, you need to specify the `-view` parameter.

`-list_non_annotated`	Lists the timing checks that do not have annotated delay information. The software checks for the following timing checks: • setup • hold • recovery • removal • pulse-width • nochange-setup • clock-period • skew **Note**: If the software is in multi-mode multi-corner timing analysis mode, you need to specify the `-view` parameter.
`-max_line` *number*	Controls the number of output lines reported by the `-list_annotated` or `list_non_annotated` parameter.
`-tcl_list`	Produces the report in Tcl list format instead of a tabular format. This is useful for integrating timing with custom Tcl functions, and also for customizing report generation. The `-tcl_list` and *`filename`* parameters are mutually exclusive; you cannot specify them together.
`-view` *viewName*	Reports coverage of annotated timing checks for the specified analysis view. You can specify this parameter only when the software is in multi-mode multi-corner timing analysis mode. *Default:* Reports annotated timing checks for the default analysis view.

Examples

- The `report_annotated_check` command shows the following report output in simultaneous setup and hold analysis mode:

```
> report_annotated_check

-----------------------------------------------------------------

| Check arc type         | Total | Annotated | Not       |Percentage|
|                        |       |           | Annotated |Arcs      |
|                        |       |           |           |Annotated |
|------------------------+-------+-----------+-----------+----------|
| Setup arcs             | 1642  | 2         | 1640      | 0.121803 |
| Hold arcs              | 1642  | 1         | 1641      | 0.060901 |
| Recovery arcs          | 368   | 0         | 368       | 0.000000 |
| Removal arcs           | 367   | 1         | 366       | 0.272480 |
| Min pulse width arcs   | 1403  | 0         | 1403      | 0.000000 |
| Period arcs            | 6     | 0         | 6         | 0.000000 |
| Clock gating hold arcs | 2     | 0         | 2         | 0.000000 |
+------------------------+-------+-----------+-----------+----------+
| All arcs               | 5430  | 4         | 5426      | 0.073665 |
+-----------------------------------------------------------------+
```

- The following command lists down all the annotated checks in the `f_slow` view:

```
> report_annotated_check -list_annotated -view f_slow
```

```
-----------------------------------------------------
Analysis View: f_slow

+-------------------------------------------------------------------------------------------
---------------+
| Setup Checks |                                  |                          |
|------------------------+-----------------------+--------------------------+-----------
---------------|
| SourceSignal         | RefSignal            | Delays            | Type
 |Cond      |View   |
|--------------------------------+--------------------------------+-------------------+---
----------------
| DTMF_IN/TDSP_CR/PROG| DTMF_IN/TDSP_CR/PROG| 0.500:0.500:0.500|
setup_rising_clk_rise|RN_EQ_1_AN|f_slow |
| BUS_MC_IN/dt_reg_2  | BUS_MC_IN/dt_reg_2  |                  |                          |
              |
| D ^                 | CK ^                |                  |                          |
```

```
      |       |
| DTMF_IN/TDSP_CR/PROG| DTMF_IN/TDSP_CR/PROG|
0.500:0.500:0.500|setup_falling_clk_rise|RN_EQ_1_AN|f_slow |
| BUS_MC_IN/dt_reg_2  | BUS_MC_IN/dt_reg_2  |                 |                       |
      |       |
| D v                 | CK ^                |                 |                       |
      |       |
+-------------------------------------------------------------------------------------
---------------

+-------------------------------------------------------------------+
| Check arc type        | Total | Annotated | Not       | Percentage|
|                       |       |           | Annotated | Arcs      |
|                       |       |           |           | Annotated |
|-----------------------+-------+-----------+-----------+-----------|
| Setup arcs            | 1642  | 2         | 1640      | 0.121803  |
| Recovery arcs         | 368   | 0         | 368       | 0.000000  |
| Min pulse width arcs | 1403  | 0         | 1403      | 0.000000  |
| Period arcs           | 6     | 0         | 6         | 0.000000  |
+-----------------------+-------+-----------+-----------+-----------+
| All arcs              | 3419  | 2         | 3417      | 0.058497  |
+-------------------------------------------------------------------+
```

- The following command lists down all the annotated checks in the f_slow view:

```
> report_annotated_check -list_annotated -view f_slow
```

```
-------------------------------------------------
Analysis View: f_fast
+-------------------------------------------------------------------------------------
---------------+
| Hold Checks |                                  |                         |
|-------------------------+-----------------------+-------------------------+-----------
---------------|
| SourceSignal        | RefSignal           | Delays           | Type
 |Cond      |View   |
|-----------------------------+-------------------------------+------------------+---
----------------
| DTMF_INST/TEST_INST | DTMF_INST/TEST_INST | 0.400:0.400:0.400| hold_rising_clk_rise |
      | f_fast|
| RAM_INST/A[5] ^     | RAM_INST/CLK ^      |                  |                       |
      |       |
| DTMF_INST/TEST_INST | DTMF_INST/TEST_INST | 0.400:0.400:0.400  hold_falling_clk_rise|
      | f_fast|
| RAM_INST/A[5] v     | RAM_INST/CLK ^      |                  |                       |
```

```
      |       |
+-------------------------------------------------------------------------------------------
--------------+

+-------------------------------------------------------------------------------------------
--------------+
| Removal Checks |                                  |                          |
|------------------------+------------------------+--------------------------+-----------
--------------|
| SourceSignal         | RefSignal            | Delays          | Type
 |Cond      |View    |
|------------------------------+------------------------------+------------------+---
--------------|

| DTMF_IN/SPI_IN/dflag| DTMF_IN/SPI_IN/dflag|0.300:0.300:0.300|removal_rising_clk_rise|
     | f_fast |
| N ^                 | K ^                  |                 |                          |
     |        |
+-------------------------------------------------------------------------------------------
--------------+
+-------------------------------------------------------------------+

| Check arc type         | Total | Annotated | Not       | Percentage|
|                        |       |           | Annotated | Arcs      |
|                        |       |           |           | Annotated |
|----------------------+-------+-----------+-----------+-------------|
| Hold arcs              | 1642  | 1         | 1641      | 0.060901  |
| Removal arcs           | 367   | 1         | 366       | 0.272480  |
| Clock gating hold arcs | 2     | 0         | 2         | 0.000000  |
+------------------------+-------+-----------+-----------+-----------+
| All arcs               | 2011  | 2         | 2009      | 0.099453 |
+-------------------------------------------------------------------+
```

Related Commands

- set_annotated_check
- reset_annotated_check

report_annotated_delay

```
report_annotated_delay
[-help]
[-ignore_tied_net_arcs]
[-list_annotated]
[-list_non_annotated]
[-view viewName]
[-max_line number]
[{> | >>} filename[.gz] | -tcl_list]
[pin_port_list]
```

Reports coverage of annotations on a design. These annotations include SDF delay arc annotations. Besides SDF annotations, the command also reports annotations that are set using the `set_annotated_delay` command.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `report_annotated_delay` parameter. For a detailed description of the command and all of its parameters, use the man command: `man report_annotated_delay`.

`> | >>` *`filename`*	When specified with >, writes the report to the specified file name. If the file already exists, the software overwrites it. When specified with >>, writes the report to the specified file name. If the file already exists, the software concatenates the report to the end of the file. The *`filename`* parameter must be the last argument in the list. The *`filename`* and `-tcl_list` parameters are mutually exclusive; you cannot specify them together. **Note:** To write a compressed report, add the `.gz` extension to the file name. *Default:* Report is displayed on standard output without being saved.
`-ignore_tied_net_arcs`	Ignores timing arcs of power and ground nets.
`-list_annotated`	Lists detailed annotation information for the following annotated arcs: • annotated internal net arcs • annotated input port arcs • annotated output port arcs • annotated cell arcs
`-list_non_annotated`	Lists the arc information for the non-annotated arcs corresponding to the following types of arcs: • non-annotated internal net arcs • non-annotated input port arcs • non-annotated output port arcs • non-annotated cell arcs
`-max_line` *`number`*	Controls the number of output lines reported by `-list_annotated` or `-list_non_annotated` parameter.

`-tcl_list`	Produces the report in Tcl list format instead of a tabular format. This is useful for integrating timing with custom Tcl functions, and also for customizing report generation. The `-tcl_list` and *`filename`* parameters are mutually exclusive; you cannot specify them together.
`-view` *`viewName`*	Generates the report for the specified analysis view only. You can specify this parameter only when the software is in multi-mode multi-corner timing analysis mode.

Examples

If your SDF has only cell arcs, and net delays are included in the cell delays, you can report the status of annotation for cell arcs, internal net arcs, net arcs from primary inputs, and net arcs to primary outputs, and all timing arcs separately. The following report is generated for `report_annotated_delay` for delay arcs types:

```
------------------------------------------------------------------------
Delay Arc Type     Total   Annotated  Not Annotated   Percentage Annotated
------------------------------------------------------------------------
Cell arcs          75      0          75              0.000000
Input net arcs     16      0          16              0.000000
Internal net arcs  59      0          59              0.000000
Output net arcs    6       0          6               0.000000
------------------------------------------------------------------------
All timing arcs    156     0          156             0.000000
------------------------------------------------------------------------
```

Related Commands

- `set_annotated_delay`
- `report_nets`

report_annotated_parasitics

```
report_annotated_parasitics
[-help]
[-include_nets <string>]
[-include_nets_file <string>]
[-view {viewName}]
[> {write_fileName}]
[>> {append_fileName}]
[ { [-list_annotated][-list_extra_pin_net] [-broken_nets ] [-floating_nets ] [-
no_driver_nets ] [-unloaded_nets ]
     [-list_not_annotated] [-list_zero_cap_net] [-real_nets ] [-supply_nets ] [-
max_missing {max_missing_int}]}]
```

Reports the back-annotated parasitics of the design. By default only summary report is generated. The software groups all the nets by net-types and annotation-status (annotated or not-annotated).

Parameters

`-help`	Prints out the command usage.
`>> {append_fileName}`	Writes the report to the specified file name. If the file already exists, the software concatenates the report to the end of the file. **Note:** The file name must be the last argument in the list. *Default:* Report is displayed on standard output without being saved.
`-help`	Outputs a brief description that includes type and default information for the `report_annotated_parasitics` parameters. For a detailed description of the command and all of its parameters, use the `man` command: `man report_annotated_parasitics`.
`-broken_nets`	Reports broken nets in the design.

`-floating_nets`	Generates a list of floating nets.
`-include_nets_file` *string*	Accepts a file name, which contains a list of net names, as input. When this parameter is specified, the parasitic annotation summary is provided only for the listed nets. The net names in the file can be specified as one net per line.
`-include_nets` *string*	Specifies a list of net names for which annotation summary is required.
`-list_annotated`	Generates a list of annotated nets.
`-list_extra_pin_net`	Generates a list of nets that have extra pins in the SPEF but not in the Verilog.
`-list_not_annotated`	Generates a list of not annotated nets.
`-list_zero_cap_net`	Generates a list of zero capacitance nets.
`-no_driver_nets`	Lists all the terminal nets that have no driver.
`-real_nets`	Lists all real nets.
`-supply_nets`	Lists all supply nets.
`-max_missing {`*max_missing_int*`}`	Limits the number of missing annotations to the number specified. *Default*: 1000
`-unloaded_nets`	Lists all the terminal nets that have no load.
`-view` *viewname*	Reports the back-annotated parasitics for the specified analysis view. You can specify this parameter only when the software is in multi-mode multi-corner timing analysis mode. *Default*: Reports against the default analysis view
`> {`*write_fileName*`}`	Writes the report to the specified file name. If the file already exists, the software overwrites it.

Examples

- The following command generates a list of annotated and not-annotated nets:

```
report_annotated_parasitics -list_annotated -list_not_annotated -list_real_net -
list_float_net -list_nodriver_net \ + -list_noload_net -list_supply_net -max_missing
1000

# list syntax: type: net_name [nrCap Cap nrXCap XCap nrRes Res]
# unit: pF, Ohm
# List the first 82 of total 82 real nets:
Annotated Real Net: [seg3/w7] [ 10 0.0690723 25 0.0512442 9 219.251 ]
Annotated Real Net: [seg3/w5] [ 14 0.0373353 24 0.0265601 13 161.843 ]
:
:
Annotated Real Net: [CLK3] [ 2 0.00046867 0 0 1 18.872 ]
Annotated Real Net (broken): [CLK4] [ 1 0.0002379 0 0 2 0.0002 ]
# List the first 20 of total 20 not annotated real nets:
Not Annotated Real Net: [seg3/wout]
Not Annotated Real Net: [CLK44]
:
:
Not Annotated Real Net: [out1]
Not Annotated Real Net: [in1]
# No annotated supply net.
# List the first 4 of total 4 not annotated supply nets:
Not Annotated Supply Net: [gnd]
Not Annotated Supply Net: [GND]
Not Annotated Supply Net: [VDD]
Not Annotated Supply Net: [vdd]
# No annotated floating net.
# No not annotated floating net.
# No annotated no driver net.
# ist the first 9 of total 9 not annotated no driver dangling nets:
Not Annotated Dangling Net (no driver): [seg3/n8]
Not Annotated Dangling Net (no driver): [seg3/n111111]
:
:
Not Annotated Dangling Net (no driver): [in]
Not Annotated Dangling Net (no driver): [out5]
# List the first 8 of total 8 annotated no load dangling nets:
Annotated Dangling Net (no load): [seg3/n14] [ 1 0.00520893 0 0 0 0 ]
Annotated Dangling Net (no load): [seg2/n14] [ 1 0.00636874 0 0 0 0 ]
:
:
```

```
Annotated Dangling Net (no load): [n6] [ 2 0.1086 0 0 1 0.0001 ]
Annotated Dangling Net (no load): [n5] [ 2 0.1065 0 0 1 0.0001 ]
# List the first 16 of total 16 not annotated no load dangling nets:
Not Annotated Dangling Net (no load): [seg3/u14:QN]
Not Annotated Dangling Net (no load): [seg3/u9:QN]
:
:
Not Annotated Dangling Net (no load): [u2:QN]
Not Annotated Dangling Net (no load): [in4]
# Summary of Annotated Parasitics:
+-------------------------------------------------------------------+
| Net Type             | Count  | Annotated (%) | Not Annotated (%) |
+----------------------+--------+---------------+-------------------+
| total                | 135    | 90 66.7%      | 45 33.3%          |
+----------------------+--------+---------------+-------------------+
| no drive (*)         | 9      | 0 0.0%        | 9 100.0%          |
| 1-term:no load       | 24     | 8 33.3%       | 16 66.7%          |
| real net (complete)  | 81     | 61 75.3%      | 20 24.7%          |
| real net (broken)    | 21     | 21 100.0%     | 0 0.0%            |
+-------------------------------------------------------------------+
Note for Net Types marked "(*)": Such nets are never timed, but reported here for
informational purpose.

+-------------------------------------------------+
| Annotated  | Res (KOhm) | Cap (pF) | XCap (pF) |
+------------+------------+----------+-----------+
| Count      | 982        | 1011     | 1152      |
| Value      | 20.0869    | 7.7289   | 5.4598    |
+-------------------------------------------------+
```

Related Commands

- read_sdf
- read_spef

report_aocv_derate

```
report_aocv_derate
[-help]
[-check_global_derating]
[-clock]
[-data]
[-list_cell_not_annotated]
[-mesh]
-out_file string
[-voltage float]
[-design | -cell string ] [ {-delay_corner <string> [-power_domain string] } | -max | -
min]
[-early | -late]
[-rise | -fall]
```

Reports the AOCV derating tables from AOCV libraries. The command generates the output report in AOCV library format.

Parameters

`-help`	Prints out command usage.
`-cell` *string*	Reports the table for the specified library cell name. You can specify a regular expression corresponding to library cell names.
`-check_global_derating`	Reports the list of cells for which derating table or global derating table was not specified.
`-clock`	Reports derate tables read for clock paths.
`-delay_corner` *string*	Reports the derating tables from the specified delay corner. For a MMMC design, if a delay corner specific table is read using the `create_library_set -aocv` command, that derating table is reported. Otherwise the default table is reported. For a non-MMMC design, no details are reported. The `-delay_corner` and `-min`/`-max` parameters are mutually exclusive.
`-data`	Reports derate tables read for data paths.

`-design`	Reports the default cell or net derating tables.
`-early` | `-late`	Reports the early/late derating table.
`-list_cell_not_annotated`	Reports the list of cells for which derating table is not specified. This parameter reports an error when specified with `-voltage`, `-design`, or `-cell` parameters.
`-max`	Reports derate tables from max AOCV libraries.
`-mesh`	Reports derate tables read for mesh.
`-min`	Reports derate tables from min AOCV libraries.
`-out_file` *string*	Specifies the name of the output file. When this parameter is specified, all the derate tables will be reported.
`-power_domain` *string*	Reports derate tables from the specified power domain. This option should always be specified with delay corner.
`-rise` | `-fall`	Reports the rise/fall table for specified transition.
`-voltage` *float*	Reports derating tables for the specified voltage value.

Examples

Here are some examples.

- The following command reports derating tables for the specified cells:

```
report_aocv_derate -cell "BUF" -delay_corner default_emulate_delay_corner -out_file
report.lib

#content of file "report.lib" :

Cell BUF
LATE-DATA-RISE
Stage 1.000000 2.000000 3.000000 4.000000 5.000000 8.000000 9.000000 18.000000
19.000000
1.180000 1.160000 1.140000 1.120000 1.100000 1.080000 1.060000 1.040000 1.020000

Cell BUF
```

```
LATE-DATA-FALL
Stage 1.000000 2.000000 3.000000 4.000000 5.000000 8.000000 9.000000 18.000000
19.000000
1.180000 1.160000 1.140000 1.120000 1.100000 1.080000 1.060000 1.040000 1.020000

Cell BUF
LATE-CLOCK-RISE
Stage 1.000000 2.000000 3.000000 4.000000 5.000000 8.000000 9.000000 18.000000
19.000000
1.180000 1.160000 1.140000 1.120000 1.100000 1.080000 1.060000 1.040000 1.020000

Cell BUF
LATE-CLOCK-FALL
Stage 1.000000 2.000000 3.000000 4.000000 5.000000 8.000000 9.000000 18.000000
19.000000
1.180000 1.160000 1.140000 1.120000 1.100000 1.080000 1.060000 1.040000 1.020000
```

- The following command reports derating tables for the cells:

```
report_aocv_derate -clock -out_file aocv_clock_derate.rpt

Cell AN2D1
EARLY-CLOCK-RISE
Stage 1.000000 2.000000 3.000000 4.000000 5.000000 6.000000 7.000000 8.000000
9.000000 10.000000
0.70000 0.850000 0.880000 0.890000 0.900000 0.910000 0.920000 0.9200000 0.930000
0.935000

Cell AN2D1
EARLY-CLOCK-FALL
Stage 1.000000 2.000000 3.000000 4.000000 5.000000 6.000000 7.000000 8.000000
9.000000 10.000000
0.760090 0.830010 0.860060 0.880300 0.890200 0.901000 0.902000 0.910000 0.920000
0.9202594
Cell BUFF
EARLY-CLOCK-RISE
Stage 1.000000 2.000000 3.000000 4.000000 5.000000 6.000000 7.000000 8.000000
9.000000 10.000000
0.842000 0.888258 0.908854 0.921954 0.929953 0.935137 0.940946 0.944305 0.942031
0.959500

Cell BUFF
EARLY-CLOCK-FALL
Stage 1.000000 2.000000 3.000000 4.000000 5.000000 6.000000 7.000000 8.000000
9.000000 10.000000
0.739532 0.810305 0.840208 0.861346 0.880605 0.899462 0.903028 0.900408 0.912154
0.916708
* The following command reports derating tables for the cells in delay corner in mesh
```

```
report_aocv_derate -mesh -delay_corner dc_125c_rcworst -out_file
aocv_clock_delay_corner_dc_125c_rcworst.rpt
Cell AN2D1
EARLY-MESH-RISE

Cell AN2D1
EARLY-MESH-FALL
Stage 1.000000 2.000000 3.000000 4.000000 5.000000 6.000000 7.000000 8.000000
9.000000 10.000000
0.739532 0.810305 0.840208 0.861346 0.880605 0.899462 0.903028 0.900408 0.912154
0.916708

Cell AN2D1
LATE-MESH-RISE
Stage 1.000000 2.000000 3.000000 4.000000 5.000000 6.000000 7.000000 8.000000
9.000000 10.000000
0.842000 0.888258 0.908854 0.921954 0.929953 0.935137 0.940946 0.944305 0.942031
0.959500

Cell AN2D1
LATE-MESH-FALL
Stage 1.000000 2.000000 3.000000 4.000000 5.000000 6.000000 7.000000 8.000000
9.000000 10.000000
0.842000 0.888258 0.908854 0.921954 0.929953 0.935137 0.940946 0.944305 0.942031
0.959500
```

Related Commands

- timing_aocv_derate_mode

report_case_analysis

```
report_case_analysis
[-all]
[-no_split]
[-dir]
[-view view_name]
[-propagated]
[-verbose]
[pins_and_ports_list]
[{> | >>} filename | -tcl_list]
```

Generates a User Case Analysis table, which reports ports and pin that have a set_case_analysis constraint. These ports and pins are constant only for timing analysis. You can optionally generate a Logic Constant Value table, which reports ports and pins whose constant status is part of the design. Information from this table includes ports or pins with `set_logic` constraints (`set_logic_zero` and `set_logic_one`), netlist explicit constants, and output pins with constant output functions.

Parameters

`{> | >>} filename`	When specified with `>`, writes the report to the specified file name. If the file already exists, the software overwrites it. When specified with `>>`, writes the report to the specified file name. If the file already exists, the software concatenates the report to the end of the file. The *`filename`* parameter must be the last argument in the list. The *`filename`* and `-tcl_list` parameters are mutually exclusive; you cannot specify them together. **Note:** To write a compressed report, add the `.gz` extension to the file name. *Default:* Report is displayed on standard output without being saved.

`-all`	Specifies that the Logic Constant Value table report pins or ports with `set_logic` constants, constants in the netlist, and output pins with a constant function. These output pins are from library cells or netlist modules.
`-no_split`	Specifies that reports with long names will not be split into multiple lines. **Note:** The `-no_split` parameter works if `set_table_style -no_frame_fix_width` command is specified.
`-dir`	Specifies that the report include the direction of the pin or port, which is reported as input, output, or inout (internal pins may not have constant constraints).
`pins_and_ports_list`	Generates a report only for the pins and ports in the specified list. *Default:* Generates a report for all pins and ports in the design.
`-propagated`	Generates a report that includes all constant pins, including pins with propagated constants. When specified, the report column header for constant values changes to `Constant value`. *Default:* Generates a report that includes only pins with a `set_case_analysis` constraint. (The column header for the constant values is labeled as `User case analysis value`.)
`-tcl_list`	Produces the report in Tcl list format instead of a tabular format. This is useful for integrating timing with custom Tcl functions, and also for customizing report generation. The `-tcl_list` and *`filename`* parameters are mutually exclusive; you cannot specify them together.

`-verbose`	Generates a report in which each pin is listed on a separate line with a cause. If the pin is a propagated constant, a table is included that reports a chain of constants from an original cause to the reported pin. *Default:* All pins are reported in a table.
`-view` *view_name*	Generates a User Case Analysis table for the specified analysis view only. To specify this parameter, the software must be in multi-mode multi-corner (MMMC) analysis mode. *Default:* When in MMMC analysis mode, generates a table for all active analysis views

Examples

- The following command uses the `-dir` option, which shows that 1 is on the input part of the `timing_const_on_Z/Z` pin:

  ```
  report_case_analysis -dir
  ```

  The following report is generated:

  ```
  ------------------------------------------------
  Pin name        Dir               User case
                                   Analysis value
  ------------------------------------------------
  gscanmode       input             0
  gscanenable     input             0
  ------------------------------------------------
  ```

- The following command generates a report that includes all constant pins, including pins with propagated constants. When specified, the report column header for constant values changes to `Constant value`.

  ```
  report_case_analysis -propagated
  ```

  The following report is generated:

  ```
  ----------------------------------------
  Pin name                   Constant value
  ```

```
-----------------------------------------
clockandlatch/U8/TE        0
clockandlatch_dup1/U8/TE   1
clockandlatch_dup2/U8/TE   1
clockandlatch_dup3/U8/TE   0
-----------------------------------------
```

- The following command generates a report in which each pin is listed on a separate line. The report includes all constant pins, including pins with propagated constants. The pin `clockandlatch_dup3/U8/TE` is reported because of a `set_case_analysis 0` assertion on pin `gscanenable`.

  `report_case_analysis -propagated -verbose clockandlatch_dup3/U8/TE`

  The following report is generated:

```
-----------------------------------------------
Pin name                       Constant value
-----------------------------------------------
gscanenable                   0
FE_OFC306_gscanenable/I       0
FE_OFC306_gscanenable/ZN      1
FE_OFC307_gscanenable/I       1
FE_OFC307_gscanenable/ZN      0
FE_OFC315_gscanenable/I       0
FE_OFC315_gscanenable/Z       0
coreclockandlatch_dup3/U8/TE  0
-----------------------------------------------
```

- The following report shows that there are no `set_case_analysis` constraints but there are logical constraints:

  `report_case_analysis -propagated -verbose core_q_reg_1_/SI`

```
-------------------------------------------
Pin name              Logic constant value
-------------------------------------------
ram256x10/TEST1       0
ram1024x32/TEST1      0
core_q_reg_2_/SI      1
core_q_reg_11_/SI     0
core_q_reg_1_/SI      1
-------------------------------------------
```

Related Commands

- set_case_analysis

report_cell_instance_timing

```
report_cell_instance_timing
[-help]
hinst_list
[-clock_timing]
[-tcl_list]
[-view view_name]
[> file_name]
[>> file_name]
[-early | -late ]
```

Reports the following information about the timing context for all the pin and delay arcs of the specified instances:

- Arrival time
- Required time
- Slack
- Delays

Note: Internal pins (pins inside the instance boundary) are also reported.

This command uses the results of earlier timing analysis runs; therefore, you must run the `report_timing` command before using `report_cell_instance_timing`.

To analyze the setup (maximum) paths, you must use the `timing_analysis_check_type setup` attribute and run the `report_timing` command before using the `-late` parameter. Similarly, to analyze the hold (minimum) paths, you must use the `timing_analysis_check_type hold` attribute and run the `report_timing` command before using the `-early` parameter.

The `report_cell_instance_timing` timing report contains the following tables:

- The first table in the report contains information about the pins of the instance.
  The propagated slew is the slew that is propagated to the pin, based on the slew propagation mode.
- The second table contains information about the arcs through the instance.
  The slew out is the slew calculated at the arc output from the arc input slew, using the timing model.
- The third table, generated for sequential devices, contains information about the timing checks, if present, on the cell.
  Only checks that are from the library are shown in this table. Automatic checks, such as clock

gating setup and hold, are not reported.

The *Phase* column of each table can contain the following symbols:

- `D` indicates it is a data signal in the clock domain.
- `C` indicates it is a clock signal in the clock domain.
- `P` indicates it is a positive phase.
- `N` indicates it is a negative phase.
- `L` indicates it is a latency phase.
- Asterisk (`*`) indicates there is a timing exception associated with the signal.

  Note: Currently, it is not possible to expand the asterisk to determine which exception is associated with the reported timing phase.

When the software is in multi-mode multi-corner timing analysis mode, the timing context information is reported for each active analysis view, with the view name listed in the *Phase* column.

Parameters

`-help`	Prints out the command usage.
`{> | >>} filename`	When specified with `>`, writes the report to the specified file name. If the file already exists, the software overwrites it. When specified with `>>`, writes the report to the specified file name. If the file already exists, the software concatenates the report to the end of the file. The *`filename`* parameter must be the last argument in the list. The *`filename`* and `-tcl_list` parameters are mutually exclusive; you cannot specify them together. **Note:** To write a compressed report, add the `.gz` extension to the file name. *Default:* Report is displayed on standard output without being saved.

`-clock_timing`	Specifies to report slack for clock style checks.
`-early` | `-late`	Reports the information for early (maximum) path (data hold/clock setup) or late (minimum) path (data setup/clock hold). *Default*: If you do not specify any of the parameters, the software uses the `-late` parameter.
hinst_list	Specifies a flat (non-hierarchical) cell instance or list of cell instances. Use curly braces (`{}`) to enclose the list and separate the instance names with white space.
`-tcl_list`	Produces the report in Tcl list format instead of a tabular format. This is useful for integrating timing with custom Tcl functions, and also for customizing report generation. The `-tcl_list` and *filename* parameters are mutually exclusive; you cannot specify them together.
`-view` *view_name*	Specifies the analysis view in which to query the object attribute.

Examples

- The following example reports information about the timing context of instance `seg1/u9`:

  Example: report_cell_instance_timing seg1/u9

```
------------------------------------------------------------------------
                   Instance seg3/u14 of DFF
------------------------------------------------------------------------
Pin   Dir   Propagated  Arrival  Required  Slack  Phase
            Slew
------------------------------------------------------------------------
CK ^ IN    0.062        0.621                    CLK_W_3(C)(P) *
CK v IN    0.065        1.503                    CLK_W_3(C)(P) *
CK ^ IN    0.062        0.621                    CLK_W_3(D)(P) *
CK v IN    0.065                                 CLK_W_3(D)(P) *
CK ^ IN    0.062                                 CLK_W_3(D)(N) *
CK v IN    0.065        1.503                    CLK_W_3(D)(N) *
```

```
D ^  IN    1.077        2.062   2.434     0.372   CLK_W_3(D)(P)
D v  IN    0.003        1.815   2.465     0.650   CLK_W_3(D)(P)
Q ^  OUT   0.044        0.788                     CLK_W_3(D)(P)
Q v  OUT   0.040        0.748                     CLK_W_3(D)(P)
QN ^ OUT   0.058        0.816                     CLK_W_3(D)(P)
QN v OUT   0.041        0.857                     CLK_W_3(D)(P)
-----------------------------------------------------------------
-----------------------------------------------------------------
               Instance seg3/u14 of DFF
-----------------------------------------------------------------
Arc         Slew
-----------------------------------------------------------------
From  To   In     Out    Load   Delay  Phase
-----------------------------------------------------------------
CK ^  QN ^ 0.062  0.058  0.000  0.196  CLK_W_3(C)(P) *
CK ^  QN v 0.062  0.041  0.000  0.236  CLK_W_3(C)(P) *
CK ^  QN ^ 0.062  0.058  0.000  0.196  CLK_W_3(D)(P) *
CK ^  QN v 0.062  0.041  0.000  0.236  CLK_W_3(D)(P) *
CK ^  QN ^ 0.062         0.000         CLK_W_3(D)(N) *
CK ^  QN v 0.062         0.000         CLK_W_3(D)(N) *
CK ^  Q ^  0.062  0.044  0.000  0.167  CLK_W_3(C)(P) *
CK ^  Q v  0.062  0.040  0.000  0.128  CLK_W_3(C)(P) *
CK ^  Q ^  0.062  0.044  0.000  0.167  CLK_W_3(D)(P) *
CK ^  Q v  0.062  0.040  0.000  0.128  CLK_W_3(D)(P) *
CK ^  Q ^  0.062         0.000         CLK_W_3(D)(N) *
CK ^  Q v  0.062         0.000         CLK_W_3(D)(N) *
-----------------------------------------------------------------
-----------------------------------------------------------------
                            Instance seg3/u14 of DFF
-----------------------------------------------------------------
            Pins                              Phase
-----------------------------------------------------------------
Check  Sig  Ref   Slack   Delay   Adjust  Sig            Ref
-----------------------------------------------------------------
HOLD   D ^  CK ^  1.570   -0.129  0.000   CLK_W_3(D)(P)  CLK_W_3(C)(P) *
HOLD   D v  CK ^  1.222   -0.027  0.000   CLK_W_3(D)(P)  CLK_W_3(C)(P) *
SETUP  D ^  CK ^  0.372   0.186   -2.000  CLK_W_3(D)(P)  CLK_W_3(C)(P) *
SETUP  D v  CK ^  0.650   0.155   -2.000  CLK_W_3(D)(P)  CLK_W_3(C)(P) *
-----------------------------------------------------------------
```

- If a constant is asserted or propagated to the instance pin, it is reported in the *Phase* column of the pin table, as shown in the following example:

Example: report_cell_instance_timing p5042A

```
-------------------------------------------------------------------
Instance p5042A of MUX2HVTD1
-------------------------------------------------------------------
Pin   Dir  Propagated   Arrival  Required  Slack    Phase
Slew
-------------------------------------------------------------------
I0 ^ IN    0.032        0.127    8.583    8.455     EJ_TCK(D)(P)
I0 v IN    0.021        0.127    8.568    8.441     EJ_TCK(D)(P)
I1 ^ IN    0.060        0.295                       EJ_TCK(D)(P)
I1 v IN    0.035        0.260                       EJ_TCK(D)(P)
S    IN                                            Constant 0
Z ^  OUT   0.020        0.168    8.624    8.455     EJ_TCK(D)(P)
Z v  OUT   0.021        0.170    8.612    8.441     EJ_TCK(D)(P)
-------------------------------------------------------------------

-------------------------------------------------------------------
Instance p5042A of MUX2HVTD1
-------------------------------------------------------------------
Arc         Slew
-------------------------------------------------------------------
From  To    In      Out    Load    Delay    Phase
-------------------------------------------------------------------
I0 ^  Z ^   0.032   0.020  0.002   0.041    EJ_TCK(D)(P)
I0 v  Z v   0.021   0.021  0.002   0.044    EJ_TCK(D)(P)
I1 ^  Z ^   0.060   0.002                   EJ_TCK(D)(P)
I1 v  Z v   0.035   0.002                   EJ_TCK(D)(P)
-------------------------------------------------------------------
```

- When the software is in multi-mode multi-corner analysis mode, the timing context information is reported for each active analysis view. The view name is reported in the *Phase* column, as shown in the following example:

```
report_cell_instance_timing FE_OFC304_EJ_TDO

-------------------------------------------------------------------------------------------
---------
                          Instance FE_OFC304_EJ_TDO of BUFFHVTD8
-------------------------------------------------------------------------------------------
---------
Pin     Dir     Propagated     Arrival         Required     Slack        Phase
                  Slew
-------------------------------------------------------------------------------------------
---------
I ^      IN      0.052         8.812           16.260    7.448     EJ_TCK(D)(N)
(setup_view1)*
```

```
I v      IN      0.033         8.799           16.319    7.520     EJ_TCK(D)(N)
(setup_view1)*
Z ^      OUT     0.338         8.919           16.367    7.448     EJ_TCK(D)(N)
(setup_view1)*
Z v      OUT     0.167         8.852           16.372    7.520     EJ_TCK(D)(N)
(setup_view1)*
-------------------------------------------------------------------------------
---------
```

Related Information

- report_nets
- report_timing

report_clock_gating_check

```
report_clock_gating_check
[obj_list]
[-view view_name]
[{> | >>} file_name]
```

Reports information of all or specific clock gating checks of the current design. If you specify a list of pins, cells, or clocks, the clock gating check information is reported for the specified objects only. If you use the `report_clock_gating_check` command without an object list, all clock gating check information is reported.

The `Type` column of the generated report (see examples below) can contain one of the following abbreviations:

- `I` indicates that the clock gating check is inferred.
- `L` indicates that the clock gating check is from the library (integrated clock gating check).

Set the `timing_report_precision` attribute using the `set_db` command to specify the accuracy of the result in terms of number of significant digits.

Parameters

obj_list	Specifies a list of pins, instances, or clocks. A collection of these can also be specified. • Pin: Any pin of a clock gating cell can be specified and the check information will be reported for that pin. • Instance: Any instance name of a clock gating cell can be specified and the check information will be reported for that instance. • Clock: Only a clock at a given clock (reference) pin of a cell can be specified for reporting the check information. Check information about the clocks at enable (signal) pin cannot be reported. If the *obj_list* parameter is not specified, all the clock gating check points (inferred and library) of the current design are reported.
-view *view_name*	Specifies a view for which clock gating check is to be reported. The default is worst of all the views.
{> | >>} *filename*	When specified with >, writes the report to the specified file name. If the file already exists, the software overwrites it. When specified with >>, writes the report to the specified file name. If the file already exists, the software concatenates the report to the end of the file.

Examples

The following command returns a clock gating check report for the current design:

```
report_clock_gating_check

The clock gating check report returned by this command is as follows:

+-------------------------------------------------------------------------+
```

Clock Gating Check Report								
Instance	Enable	Clock	Type	Level	Setup Rise	Setup Fall	Hold Rise	Hold Fall
i1	A	B	I	L	0.000	0.000	-	-
i2	B	A	I	H	0.000	0.000	-	-
icg	CEN	CLK	L	H	0.149	0.181	-	-

report_clock_propagation

```
report_clock_propagation
[-help]
-clock clock_list
[-max_paths number of paths]
-to pin_port_list
[-verbose]
[-view viewName]
[> filename[.gz]]
[>> filename[.gz]]
```

This command provides the ability to determine clocks that do not propagate to a particular node in the network, and report the causes for such occurrences.

Parameters

`-help`	Prints out the command usage
`-clock clock_list`	List or collection of clocks that are to be checked.
`-max_paths number of paths`	Reports number of maximum clock paths per clock.
`-to pin_port_list`	Specifies the pin/port list.
`-verbose`	Generates a verbose report for clock propagation stop reason.
`-view viewName`	Generates a report for the specified analysis view.
`> filename[.gz] | >> filename[.gz]`	When specified with >, writes the report to the specified file name. If the file already exists, the software overwrites it. When specified with >>, writes the report to the specified file name. If the file already exists, the software concatenates the report to the end of the file. **Note**: To write a compressed report, add the .gz extension to the file name.

Examples

- The following command shows that clock propagation stops due to an inactive timing arc from b12/A to b12/Y

```
set_disable_timing -from A -to Y b12
report_clock_propagation -to u99/CK -clock CLK_W_4

Clock: CLK_W_4
Point: u99/CK
Reason 1: The clock 'CLK_W_4' propagation stops due to the inactive timing arc:
(b12/A->b12/Y)

The partial clock path:

-------------------------------------------------
Source    Sink      ArcType         ArcSense
-------------------------------------------------
CLK4     b11/A   (net)           (net)
b11/A    b11/Y   combinational   positive_unate
b11/Y    b12/A   (net)           (net)
b12/A    b12/Y   combinational   positive_unate
-------------------------------------------------
```

- The following command shows that clock propagation stops due to no clock source found:

```
report_clock_propagation -to launchff/Q -clock [get_clocks gen_clk]
report_clock_propagation -to seg3/u15/Q -clock CLK_W_5_gen


Clock: CLK_W_5_gen
Point: seg3/u15/Q
Reason 1: The generated clock ‘CLK_W_5_gen could not be created on its root pin
‘seg3/u15/Q’ because no clock source found for it
```

- The following command shows that clock propagation stops at the register clock pin as all the arcs from the clock pin of the register have been disabled:

```
set_disable_timing -from A -to Y seg3/u17_a
report_clock_propagation -to seg3/u14/CK -clock CLK_W_3

Clock: CLK_W_3
Point: seg3/u14/CK
Reason 1: The clock 'CLK_W_3' propagation stops due to the inactive timing arc:
(seg3/u17_a/A->seg3/u17_a/Y)
```

```
The partial clock path:
-------------------------------------------------------------
Source          Sink          ArcType         ArcSense
-------------------------------------------------------------
CLK3           seg3/u16/A     (net)           (net)
seg3/u16/A     seg3/u16/Y     combinational   positive_unate
seg3/u16/Y     seg3/u17/A     (net)           (net)
seg3/u17/A     seg3/u17/Y     combinational   positive_unate
seg3/u17/Y     seg3/u17_a/A   (net)           (net)
seg3/u17_a/A   seg3/u17_a/Y   combinational   positive_unate
-------------------------------------------------------------
```

Related Commands

- set_disable_timing

report_clock_timing

```
report_clock_timing
[-cppr_relative]
[-latency_greater_than]
-type [skew | interclock_skew | jitter | summary | latency | cppr_stage_count
      [ [-launch | -capture]
        [-rise | -fall]
        | [-histogram [-histogram_range interval size]]
        [-logic_level [-source {clock_root | generated_clock}]]
        ]
      ]
[-absolute_compare]
[-early | -late]
[-clock clock_list]
[-from_clock from_clock_list]
[-to_clock to_clock_list]
[-from from_list]
[-to to_list]
[-nworst worst_entries]
[-greater_than lower_limit]
[-view view_name]
[-verbose]
[-format column_list]
[-tcl_list]
[{> | >>} filename]
```

Generates a clock skew report for the current design. The software generates a separate report for each specified clock or pair of clocks. You specify the clocks using the `-clock` parameter, and clock pairs using the `-from_clock` and `-to_clock` parameters. You can use the `-nworst` and `-greater_than` parameters with the `-from_clock` and `-to_clock` parameters to generate reports with skew to the specified sink pins. You can use the `-early` and `-late` parameters to specify the type of skew to be reported.

You can use the `report_clock_timing` command to perform MMMC based SSTA analysis. In MMMC mode, the command reports the latency and clock skew for all individual views. You can use the `report_clock_timing -view` parameter to report clock timing for a user-specified view.

Parameters

{> | >>} *filename*	When specified with >, writes the report to the specified file name. If the file already exists, the software overwrites it. When specified with >>, writes the report to the specified file name. If the file already exists, the software concatenates the report to the end of the file. The *filename* parameter must be the last argument in the list. The *filename* and -tcl_list parameters are mutually exclusive; you cannot specify them together. **Note:** To write a compressed report, add the .gz extension to the file name. *Default:* Report is displayed on standard output without being saved.
-clock *clock_list*	Specifies the list of clocks to be reported. The software generates a separate report for each clock that you specify in the *clock_list*. *Default:* Skew report is generated for all clocks in the design.
-cppr_relative	Reports the latency in skew reports from the CPPR common pin. When specified, the reports are printed with latency measured from branch point to the register's CK pin. The CPPR value is no longer required in reports.
-early	Uses hold skew to generate the report. Hold skew is the difference between minimum latency at the start point flip-flop and the maximum latency at the end point flip-flop. Default: Uses the setup skew.

`-format` *`column_list`*	Specifies which columns to display for path reports, and the order in which they should appear. For example: `-format {instance arc pin cell slew load}` The `-format` parameter can be used only for latency or skew reports (`-type skew`, `-type interclock_skew`, or `-type latency`), and if the `-verbose` parameter is specified.
`-from` *`from_list`*	Specifies the list of sequential device clock pins or ports to consider as from-pins in the generated report. If any pin in the *`from_list`* is not a sequential device clock pin, it is replaced with all the sequential device clock pins in the transitive fanout of the named pin. If there are no sequential pins in the transitive fanout of the pin, the software drops the pin from the list and generates a warning message.
`-from_clock` *`from_clock_list`*	Specifies the list of from-clocks to be used in generating the current interclock skew report. You can use this option only in an interclock skew report. *Default:* Report is generated considering all clocks as from-clocks.
`-greater_than` *`lower_limit`*	Reports skews with values greater than the *`lower_limit`* value.

`-histogram`	Generates a textual histogram of the latency report. The `-histogram` parameter can be used only when `-type latency` is specified. When specified in conjunction with the `-clock` parameter, the software limits the latency reporting to the specified clocks only. When specified with the `-greater_than` parameter, the software limits the latency reporting to clocks with a latency value that is greater than the specified value. **Note:** The `-histogram` parameter cannot be specified in conjunction with the following parameters: `-launch`,`-capture`, `-from`, `-to`, `-from_clock`, `-to_clock`, `-verbose`, `-nworst`.
`-histogram_range` *`interval_size`*	Generates the histogram report at the specified intervals. The interval size specified must be a positive integer or decimal values. For example, if the maximum latency value is 437, and you specify `-histogram_range 50`, the histogram would be generated with intervals of: 0-50, 50-100, 100-150, ... 400-450. **Note:** The unit of *`interval_size`* will be the same as that used for generating the timing report. This parameter can be used only when the `-histogram` parameter is specified.
`-late`	Uses setup skew to generate the report. Setup skew is the difference between the maximum latency at the start point flip-flop and minimum latency at the end point flip-flop.
`-latency_greater_than <`*`lower_limit`*`>`	Reports are filtered based on the specified -type option. When specified with: • `-type skew` or `interclock_skew`: Reports are filtered for latency greater than the provided value or depth. • `-type cppr_stage_count`: Reports are filtered for depth greater than the provided value.

`-launch	-capture`	When specified with `-type latency`, generates a report of either the launch latency values (`-launch`), or the capture latency values (`-capture`). **Note:** You can specify these parameters only with `-type latency`. *Default*: `-launch`
`-logic_level`	Reports the levels (the stage count of gates) between the clock root, or generated clock root, and the capturing register. The `-logic_level` parameter can be used only when `-type latency` is specified. The `-logic_level` parameter cannot be specified in conjunction with the following parameters: `-launch`,`-capture`, `-from`, `-to`, `-from_clock`, `-to_clock`, `-verbose`, `-nworst`.
`-nworst` *worst_entries*	Specifies the number of entries to be reported per clock domain. *Default:* `1`
`-rise	-fall`	Specifies active transition at sequential devices clock pins. The `-rise` and `-fall` parameters are mutually exclusive and cannot be specified together. **Note:** You can specify these parameters only with the `-type latency` parameter.

`-source {clock_root | generated_clock}`	Reports the latency from either the master clock, or the generated clock that was derived from it. The `-source` parameter can be used only when `-logic_level` is specified. The `-source` parameter cannot be specified in conjunction with the following parameters: `-launch`,`-capture`, `-from`, `-to`, `-from_clock`, `-to_clock`, `-verbose`, `-nworst`. *Default:* Reports the latency from the master clock (`-source clock_root`)
`-tcl_list`	Produces the report in Tcl list format instead of a tabular format. This is useful for integrating timing with custom Tcl functions, and also for customizing report generation. The `-tcl_list` and *`filename`* parameters are mutually exclusive; you cannot specify them together.
`-to` *`to_list`*	Specifies the list or collection of sequential device clock pins or ports to be considered for reporting. If any pin in the *`to_list`* is not a sequential device clock pin, it is replaced with all the sequential device clock pins in the transitive fanout of the named pin. If there are no sequential pins in the transitive fanout of the pin, the software drops the pin from the list and generates a warning message.
`-to_clock` *`to_clock_list`*	Specifies the list of to-clocks to be used in generating the current interclock skew report. You can use this option only in an interclock skew report. *Default:* Report is generated considering all clocks as to-clocks.
`-type {skew | interclock_skew | jitter | summary | latency | cppr_stage_count}`	Specifies the type of report to generate. The `-type` options are described below: • `skew`: Reports the skew for all clocks, or for clocks that you specify using the -clock

parameter.

- `interclock_skew`: Reports the skew between all clocks in the design, or between clocks that you specify using the `-from_clock` and `-to_clock` parameters.
- `jitter`: Reports the difference in late and early arrival times of a clock phase at a clock end point. When specified, the software generates a clock jitter report for one clock end point, for each clock in the design.
  You can generate a jitter report for all clocks, or for clocks specified with the `-clock` parameter. You can limit the report to a specific clock end point using the `-to` parameter.
  Note: The `report_clock_timing -type jitter` option can be used only when the software is in On-Chip Variation (OCV) timing analysis mode. Jitter analysis does not remove any common path pessimism.
- `summary`: Generates a summary report for each analyzed clock that includes the following information:
  - When setup skew is used (`-late`):
    - maximum launch latency
    - minimum capture latency
    - maximum setup skew
  - When hold skew is used (`-early`):
    - minimum launch latency
    - maximum capture latency
    - maximum hold skew
- `latency`: Reports the network and source latency values for all the clock pins in the

	design, or for clock pins that you specify using the `-clock` parameter.
	• `cppr_stage_count`: The generated report is arranged in terms of difference of stages from common pin to CK pin. The reports are generated on the basis of logical CPPR points derived between each pair of phases reaching the launch/capture CK pins. Only same clock pairs (similar to type skew) are picked. For example,
	`Branch Point Difference Depth Clock Pin`
	`----------------------------------------`
	`-`
	`inv2/Y`
	`5 r ffc1/CK`
	`4 1 f ffc3/CKN`

`-verbose`	Generates a report that contains the full clock path from the clock source to the sink pin (the clock pin of the flop or latch). You can specify the `-verbose` parameter only for skew or latency reports (`-type skew`, `-type interclock_skew`, or `-type latency`). When this parameter is specified in SOCV mode, the intermediate arrival times of pins are shown as "-" in the report output.
`-view` *`view_name`*	Generates a report for the specified analysis view only. You can specify this parameter only when the software is in multi-mode multi-corner timing analysis mode.

Examples

- The following command reports worst case skew for all clocks in a design:

```
report_clock_timing -type skew -nworst 4

  Clock: CLK_W_4
  Skew    Latency   Clock Pin
```

```
----------------------------
        0.150 r   u99/CK
0.000   0.150 r   u55/CK

Clock: CLK_W_3
Skew   Latency   Clock Pin
----------------------------
        0.682 r seg3/u9/CK
0.061   0.621 r seg3/u14/CK

Clock: CLK_W_2
Skew    Latency Clock Pin
----------------------------
        0.682 r seg2/u9/CK
0.061   0.621 r seg2/u14/CK

Clock: CLK_W_1
Skew    Latency Clock Pin
----------------------------
        0.682 r seg1/u9/CK
0.061   0.621 r seg1/u14/CK
```

Here 'r' indicates the transition (r - rise, f - fall) at the reported register clock pin.

- The following command reports interclock skew with -from_clock, -to_clock, -from, and -to parameters defined:

```
report_clock_timing -type interclock_skew -nworst 10 -from CLK4 -to u5/CK -from_clock
CLK_W_4 -to_clock CLK_W_4_gen2

  Clocks: CLK_W_4 -> CLK_W_4_gen2
  Skew       Latency Clock Pin
  ----------------------------
                     CLK4
             0.180 r u99/CK
                     u2/Q
                     u8/Q
  -0.205    0.386 ri u5/CK
```

  Here 'r' indicates the transition (r - rise, f - fall) at the reported register clock pin. The 'i' refers to the 'inverted' polarity of the clock signal at the reported register clock pin.

- The following command reports skew for all clocks in the design including common path pessimism removal:

```
set_analysis_mode -cppr both
report_clock_timing -type skew -nworst 10
```

```
Clock: CLK_W_4
Skew    Latency    CPPR    Clock Pin
---------------------------------------
        0.180              r u99/CK
0.000   0.120       -0.060 r u55/CK

Clock: CLK_W_3
Skew     Latency  CPPR    Clock Pin
---------------------------------------
         0.818            r seg3/u9/CK
0.139    0.496 -0.183     r seg3/u14/CK

Clock: CLK_W_2
Skew    Latency  CPPR     Clock Pin
---------------------------------------
        0.818             r seg2/u9/CK
0.139   0.496    -0.183    r seg2/u14/CK

Clock: CLK_W_1
Skew     Latency   CPPR     Clock Pin
---------------------------------------
         0.818               r seg1/u9/CK
0.139    0.496    -0.183     r seg1/u14/CK
```

- The following command generates a summary report for the clock (Not mentioned in mail)

```
report_clock_timing -type summary -nworst 10 -clock CLK_W_1

Clock: CLK_W_1    Skew    Latency    Clock Pin
-------------------------------------------
Maximum launch latency:   0.818 r    seg1/u9/CK
Minimum capture latency:  0.000 r    u2/CK
Maximum skew:             0.818 r    seg1/u9/CK
                  0.322   0.496 r    seg1/u14/CK
-------------------------------------------
20. The following command reports the Network and source latency values for the
clock CLK_W_1:

report_clock_timing -type latency -nworst 10 -clock CLK_W_1

Clock: CLK_W_1
---Latency---
Source  Network  Total   Clock Pin
-------------------------------------------
0.000   0.818    0.818   r seg1/u9/CK
0.000   0.000    0.000   r u2/CK
-------------------------------------------
```

- The following command reports the Network and source latency values for the clock GCLK1, as well as the full clock paths:

```
report_clock_timing -type latency -verbose 10 -clock CLK_W_1
```

```
  Clock: CLK_W_1
  ---Latency---
  Source Network Total Clock Pin
  ----------------------------------------
  0.000 0.818 0.818 r seg1/u9/CK

  Clock latency Path
  Clock Rise Edge 0.000
  = Beginpoint Arrival Time 0.000
  Other End Path:
  ----------------------------------------
  Instance       Arc      Cell     Delay   Clock
  Latency
  ------------------------------------------------
  -            CLK1 ^       -      -       0.000
  seg1/u16     A ^ -> Y ^  CLKBUF   0.111  0.111
  seg1/u17     A ^ -> Y ^  CLKBUF   0.126  0.236
  seg1/u17_a   A ^ -> Y ^  CLKBUF   0.107  0.343
  seg1/and1    B ^ -> Y ^  AND      0.071  0.818
  seg1/u9      CK ^        DFF      0.000  0.818
  ------------------------------------------------
```

- When the software is in multi-mode multi-corner timing analysis mode, the software generates a report for every active analysis view in the design:

```
report_clock_timing -type latency
```

```
  Clock: EJ_TCK
  Analysis View: hold_view1
  ---Latency---
  Source Network Total Clock Pin
  ----------------------------------------
  0.000 0.000 0.000 r cregister_q_reg_3_/CP

  Clock: EJ_TCK
  Analysis View: hold_view2
  ---Latency---
  Source Network Total Clock Pin
  ----------------------------------------
  0.000 0.000 0.000 r cregister_q_reg_3_/CP
```

```
Clock: EJ_TCK
Analysis View: setup_view1
---Latency---
Source Network Total Clock Pin
-----------------------------------------
0.000 0.000 0.000 r cregister_q_reg_3_/CP
```

- Specify the `-view` parameter to generate a report only for a specific analysis view:

```
report_clock_timing -view setup_view1 -type latency
```

```
Clock: EJ_TCK
Analysis View: setup_view1
---Latency---
Source    Network    Total    Clock       Pin
-----------------------------------------
0.000     0.000      0.000    r    cregister_q_reg_3_/CP

Clock: SI_ClkIn
Analysis View: setup_view1
---Latency---
Source   Network   Total    Clock        Pin
-----------------------------------------
0.000    0.000     0.000     r     cregister_q_reg_13_/CP
```

- The following command generates a textual histogram of the latency report:

```
report_clock_timing -type latency -histogram
```

```
[Delay Chart: CLK4(CLK_W_4_gen2)]
************************************
Min Delay : 0.778
Max Delay : 0.778
Count : 1
************************************
----------------------------------------
Delay    Range
Min      Max      Count   1
----------------------------------------
0.000  100.000 1          |o
```

- The following command generates the histogram report at intervals of `0.5`:

```
report_clock_timing -type latency -histogram -histogram_range 0.5
```

```
[Delay Chart: CLK4(CLK_W_4_gen2)]
**************************************
Min Delay : 0.778
Max Delay : 0.778
Count : 1
**************************************
--------------------------------------
Delay Range
Min    Max    Count   1
--------------------------------------
0.500  1.000  1       |o
0.000  0.500  0
```

- The following command generates a latency report that includes the stage gate counts between the clock root or generated clock root, and the capturing register:

```
report_clock_timing -type latency -logic_level
```

```
************************************************
Start point:CLK2(CLK_W_4_gen1(CLK_W_3))
-----------------------------------------------
Delay   Count   End Point
1.018   8       seg2/u9/CK r
1.018   8       seg2/u3/CK r
0.945   7       seg2/u14/CK r

************************************************
Start point:CLK3(CLK_W_4_gen2(CLK_W_4))
-----------------------------------------------
Delay   Count  End Point
1.345   8      seg3/u6/CK r
1.345   8      seg3/u1/CK r
1.218   7      seg3/u12/CK r
```

- The following command generates a latency report starting from the generated clock, and includes the stage gate counts between the generated clock and the capturing register:

```
report_clock_timing -type latency -logic_level -source generated_clock
```

```
************************************************
Start point:CLK2(CLK_W_4_gen1(CLK_W_3))
-----------------------------------------------
Delay   Count   End Point
0.818   8       seg2/u9/CK r
0.818   8       seg2/u3/CK r
0.745   7       seg2/u14/CK r
```

```
**************************************************
Start point:CLK3(CLK_W_4_gen2(CLK_W_4))
-----------------------------------------
Delay   Count   End Point
0.845   8       seg3/u6/CK r
0.845   8       seg3/u1/CK r
0.718   7       seg3/u12/CK r
```

- The following command generates a latency report starting from the generated clock CLK_W_4_gen2, and includes the stage gate counts between the generated clock and the capturing register:

```
report_clock_timing -type latency -logic_level -source generated_clock -clock
CLK_W_4_gen2
```

```
**************************************************

Start point:CLK3(CLK_W_4_gen2(CLK_W_4))
-----------------------------------------
Delay   Count    End Point
0.845   8        seg3/u6/CK r
0.845   8        seg3/u1/CK r
0.718   7        seg3/u12/CK r
```

- The following command uses the -format parameter to customize the generated latency report:

```
report_clock_timing -type latency -verbose -format {instance arc pin cell slew load
user_derate}
```

```
Clock: CLK_W_4

---Latency---
Source Network Total Clock Pin
---------------------------------------
0.000 0.180 0.180 r u8/CK

Clock latency Path
Clock Rise Edge 0.000
= Beginpoint Arrival Time 0.000
Other End Path:
-------------------------------------------------------------
Instance  Arc          Pin     Cell  Slew      Load     User
                                                        Derate
-------------------------------------------------------------
-         CLK4 ^       CLK4    -     0.003     0.024    -
b11       A ^ -> Y ^   Y       BUF   0.131     0.011    1.200
b12       A ^ -> Y ^   Y       BUF   0.116     0.013    1.200
u8        CK ^         CK      DFF   0.116     0.013    1.200
```

```
  ------------------------------------------------------------
```

- The following command reports the worst case jitter for all clocks in the design:

```
report_clock_timing -type jitter -nworst 2

****************************************
 Report: clock timing
 -type jitter
 -setup
 -nworst 2
 CRPR analysis: off
****************************************
 Clock: CLK1
 Jitter   Latency(Late)   Latency(Early)   Clock Pin
------------------------------------------------------
 0.186    0.350 r         0.164 r          i_17/CK
 0.118    0.227 r         0.109 r          i_15/CK

 Clock: CLK0

 Jitter Latency(Late)  Latency(Early)    Clock Pin
-----------------------------------------------------------
 0.387  0.508     r    0.120      r      i_5/CK
 0.178  0.336     r    0.159      r      i_7/CK
```

- The following command reports the worst case jitter for the clock CLK1:

```
report_clock_timing -type jitter -clock CLK1 -nworst 2

****************************************
 Report: clock timing
 -type jitter
 -setup
 -nworst 2
 CRPR analysis: off
****************************************
 Clock: CLK1
```

```
 Jitter   Latency(Late) Latency(Early) Clock Pin
-----------------------------------------------------
 0.186    0.350    r    0.164    r     i_17/CK
 0.118    0.227    r    0.109    r     i_15/CK
```

- The following command reports the worst case jitter for the specified clock end point:

```
report_clock_timing -type jitter -to i_15/CK

****************************************
 Report: clock timing
 -type jitter
 -setup
 -nworst 2
 CRPR analysis: off
****************************************
 Clock: CLK1

 Jitter   Latency(Late)  Latency(Early)   Clock Pin
-------------------------------------------------------------
 0.118    0.227      r   0.109      r     i_15/CK
```

Related Commands

- report_timing
- check_timing

report_clocks

```
report_clocks
[-description]
[-arrival_points]
[-clock_source_jitter]
[-phase_shift_table]
[-total_shift_table]
[-uncertainty_table]
[-adjustment_table]
[-delay_adjustment_table]
[-source_insertion]
[-insertion]
[-hierarchy]
[-groups]
[-clocks clk_signame | clk_signame_list]
[-view view_name]
[{> | >>} filename | -tcl_list]
```

Reports information on clocks created with the `create_clock` and `create_generated_clock` command. The report includes information about all clock waveforms, clock arrival points, and clock uncertainties.

Use the `-description`, `-arrival_points`, `-uncertainty_table`, and the `*_table` parameters to specify different parts of the report. If one of these options is supplied, only the section that the option corresponds to is reported. If other options are supplied, those sections are also reported.

Based on the current setup or hold analysis mode, the following parameters report only one of the late or early tables:

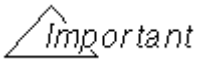
- `-phase_shift_table`
- `-total_shift_table`
- `-uncertainty_table`
- `-adjustment_table`
- `-delay_adjustment_table`

However, in simultaneous (setup and hold) mode both the tables are reported.

The `report_clocks` command only reports constraints with respect to clock waveforms. Use the `report_ports -type` command for constraints applied to clock pins.

Parameters

{> | >>} *filename*	When specified with >, writes the report to the specified file name. If the file already exists, the software overwrites it. When specified with >>, writes the report to the specified file name. If the file already exists, the software concatenates the report to the end of the file. The *filename* parameter must be the last argument in the list. The *filename* and -tcl_list parameters are mutually exclusive; you cannot specify them together. **Note:** To write a compressed report, add the .gz extension to the file name. *Default:* Report is displayed on standard output without being saved.
-adjustment_table	Reports two tables that detail the hold and setup cycle adjustments between clock waveforms as set by the set_multicycle_path command. **Note:** If specific ideal (or generated) clocks are requested with the -clocks *clock_name_list* option, then only the relationships between those clock signals are reported.
-arrival_points	Reports only the clock arrival points. A clock arrival point is the pin where the create_clock or create_generated_clock command has been asserted. **Note:** This parameter is obsolete and will be removed in a future release.

`-clocks` *`clk_signame`* | *`clk_signame_list`*	Requests information on a specific ideal (or generated) clock. If a list is given, only information about those clocks is reported. You can use wildcards in the names. *Default*: All clocks, except generated clocks, are reported. Use the `-generated` option to report generated clocks.  This parameter is obsolete and will be removed in a future release. Cadence recommends that you specify clock names without the `-clocks` parameter.
`-clock_source_jitter`	Reports the clock source jitter applied to specific clocks. When this parameter is specified, the report output shows the clock and master clock columns to indicate the inheritance of source clock jitter from a master clock (defined using the `create_clock` command) to a generated clock.
`-delay_adjustment_table`	Reports two tables that detail the early and late path delay adjustments between clock waveforms as set by the `set_max_delay` and `set_min_delay` commands. **Note:** If specific ideal (or generated) clocks are requested with the `-clocks` *`clock_name_list`* option, then only the relationships between those clock signals are reported.
`-description`	Reports only the description of the specified clock waveform(s).

`-generated`	Includes information on generated clocks in the report, such as source and master clock of the generated clock, and other information specified in the `create_generated_clock` constraints. The report uses the internal clock names created by the system, if you did not specify new clock names while creating these clocks. ⓘ This parameter is obsolete and will be removed in a future release. The software now reports generated clocks by default.
`-groups`	Reports the existing clock groups.
`-hierarchy`	Reports the inter-clock relationship in tree style format.
`-insertion`	Reports the network insertion delays specified on clock waveforms. The values in the table are in `min:typ:max` format.
`-phase_shift_table`	Reports the table that details the phase shift between clock waveforms. Based upon the current analysis mode, the table shows the phase shifts either in late or early mode. In case of simultaneous setup hold mode, both the tables are reported for phase shift in late and early mode. **Note:** If specific ideal (or generated) clocks are requested with the `-clocks` *`clock_name_list`* option, then only the relationships between those clock signals are reported.
`-source_insertion`	Reports the source insertion delays specified on clock waveforms.

`-tcl_list`	Produces the report in Tcl list format instead of a tabular format. This is useful for integrating timing with custom Tcl functions, and also for customizing report generation. The `-tcl_list` and *`filename`* parameters are mutually exclusive; you cannot specify them together.
`-total_shift_table`	Based on the current analysis mode, reports one of the tables (late or early mode) that details the total relationship between clock waveforms. In case of simultaneous setup hold mode both the tables are reported. **Note:** If specific ideal (or generated) clocks are requested with the `-clocks` *`clock_name_list`* option, then only the relationships between those clock signals are reported.
`-uncertainty_table`	Based on the current analysis mode, reports one of the tables (late mode or early mode) that detail the uncertainty between clock waveforms as specified using the `set_clock_uncertainty` command. In case of simultaneous setup hold mode both the tables are reported. The tables report only those clock waveforms for which uncertainty has been defined using the `set_clock_uncertainty` command. **Note:** If specific ideal (or generated) clocks are requested with the `-clocks` *`clock_name_list`* option, then only the relationships between those clock signals are reported.
`-view` *`view_name`*	Generates a clock report for the specified analysis view only. You can specify this parameter only when the software is in multi-mode multi-corner timing analysis mode.

Examples

Here are some examples.

- report_clocks

```
------------------------------------------------------
Clock Descriptions
------------------------------------------------------
Attributes
------------------------------------------------------------
Clock Name  Source  Period  Lead   Trail  Generated Propagated
------------------------------------------------------------
CLK1        CLK1    20.000  0.000  10.000  n        y
CLK2        CLK1    10.000  0.000   5.000  n        y
------------------------------------------------------------
Generated-Clock Descriptions
-------------------------------------------------------------------------------------
---------
Name        Generated   Master      Master-clock Invert Freq.      Duty-Cycle Edges
Edge-Shift
            Source(pin) Source(pin)                  Multiplier
-------------------------------------------------------------------------------------
---------
GCLK        u8/Q        CK1         CLK1         n      1/2         -          -
 -
-------------------------------------------------------------------------------------
--------
```

- The following command displays only the uncertainty information:

```
set_clock_uncertainty 5 CLK_W_1
set_clock_uncertainty 4 -from CLK_W_1 -to CLK_W_4_gen1
set_clock_uncertainty 6 -from CLK_W_2 -to CLK_W_4_gen2
report_clocks -uncertainty_table
------------------------------------------------------
Clock To Clock Uncertainty (Late)
```

```
-------------------------------------------------------
From    To            L->L    L->T    T->L     T->T
-------------------------------------------------------
*       CLK_W_1       5.000  5.000   5.000     5.000
CLK_W_1 CLK_W_4_gen1 4.000  4.000   4.000     4.000
CLK_W_2 CLK_W_4_gen2 6.000  6.000   6.000     6.000
-------------------------------------------------------
Clock To Clock Uncertainty (Early)
-------------------------------------------------------------
From     To             L->L   L->T   T->L   T->T
-------------------------------------------------------------
*        CLK_W_1       -5.000 -5.000 -5.000 -5.000
CLK_W_1  CLK_W_4_gen1 -4.000 -4.000 -4.000 -4.000
CLK_W_2  CLK_W_4_gen2 -6.000 -6.000 -6.000 -6.000
-------------------------------------------------------------
```

- Consider the following clock definition:
  create_clock -name "CLK_W_1" -add -period 60 -waveform {0 30}
  report_clocks CLK_W_1 -phase_shift_table

```
-----------------------------------------------------
Clock To Clock Phase Shift (Late)
-----------------------------------------------------
From    To      L->L    L->T  T->L   T->T
-----------------------------------------------------
CLK_W_1 CLK_W_1 60.000  0.000 60.000 60.000
-----------------------------------------------------
-----------------------------------------------------
Clock To Clock Phase Shift (Early)
-----------------------------------------------------
From    To      L->L   L->T   T->L  T->T
-----------------------------------------------------
CLK_W_1 CLK_W_1 0.000 -60.000 0.000 0.000
-----------------------------------------------------
```

- You can use the following command to add uncertainty:
  `set_clock_uncertainty 2 CLK_W_1`

  The phase shift and total shift table values will be different. The following command displays the changed shift values as given below:
  `report_clocks -total_shift_table CLK_W_1`

```
------------------------------------------------------
Total Clock To Clock Phase Shift Required (Late)
------------------------------------------------------
From    To      L->L    L->T  T->L   T->T
------------------------------------------------------
CLK_W_1 CLK_W_1 58.000 -2.000 58.000 58.000
------------------------------------------------------
------------------------------------------------------
Total Clock To Clock Phase Shift Required (Early)
------------------------------------------------------
From    To      L->L   L->T   T->L  T->T
------------------------------------------------------
CLK_W_1 CLK_W_1 2.000 -58.000 2.000 2.000
------------------------------------------------------
```

You can see that the uncertainty is subtracted from the phase shift. Consider the following example path:

```
Path 1: MET Setup Check with Pin u2/CK
Endpoint: u2/D (v) checked with leading edge of 'CLK_W_1'
Beginpoint: CLK2 (v) triggered by trailing edge of 'CLK_W_2'
Path Groups: {CLK_W_1}
Other End Arrival Time 0.000
- Setup 0.188
+ Phase Shift 15.000
- Uncertainty 2.000
= Required Time 12.812
- Arrival Time 12.463
= Slack Time 0.349
Clock Fall Edge 12.500
= Beginpoint Arrival Time 12.500
-----------------------------------
Load  Slew    Delay   Arrival   Cell  Arc          Pin
                      Time
-------------------------------------------------------
0.034  0.003  12.500                  CLK2 v       CLK2
0.003  0.003  -0.037  12.463   BUF    A v -> Y v   u10999/Y
0.003  0.003  0.000   12.463   DFF    D v          u2/D
-------------------------------------------------------
```

Here the begin point CLK_W_2 to end point CLK_W_1 shows a phase shift value of 15.00. This means that if a path starts from the falling edge of CLK_W_2 , and ends at the rising edge of CLK_W_1, the phase shift is 15.00. In other words, when the timing (setup time) is reported, the required time at the end point begins with 15.00 (in addition to adjustments like setup time, time borrowing, uncertainty, etc). However, the arrival time of this path starts at 12.500.

- When the software is in multi-mode multi-corner timing analysis mode, the software reports the clock information for each active analysis view:

```
report_clocks -total_shift_table -description

------------------------------------------------------------
Clock Descriptions
------------------------------------------------------------
Attributes
------------------------------------------------------------------------
Clock Name Source  View       Period  Lead   Trail Generated Propagated
------------------------------------------------------------------------
EJ_TCK     EJ_TCK  hold_view2 17.400  0.000  8.700  n         n
EJ_TCK     EJ_TCK  hold_view1 17.400  0.000  8.700  n         n
EJ_TCK     EJ_TCK  setup_view117.400  0.000  8.700  n         n
------------------------------------------------------------------------
------------------------------------------------------------
Total Clock To Clock Phase Shift Required (Late)
------------------------------------------------------------
From      To        View        L->L   L->T   T->L    T->T
------------------------------------------------------------
EJ_TCK    EJ_TCK    hold_view2  17.400  0.000 17.400  17.400
EJ_TCK    EJ_TCK    hold_view1  17.400  0.000 17.400  17.400
EJ_TCK    EJ_TCK    setup_view2 17.400  0.000 17.400  17.400
------------------------------------------------------------------------
```

- Specify the `-view` parameter to generate clock information for a specific analysis view only:

```
report_clocks -total_shift_table -description -view setup_view1 > clk_sv1

------------------------------------------------------------
Clock Descriptions
------------------------------------------------------------
Attributes
------------------------------------------------------------
Clock Name      Source  View         Period  Lead  Trail  Generated Propagated
------------------------------------------------------------------------------
EJ_TCK          EJ_TCK  setup_view1  17.400  0.000 8.700  n         n
virt_in_EJ_TCK  -       setup_view1  17.400  0.000 8.700  n         n
------------------------------------------------------------------------------
------------------------------------------------------------------
Total Clock To Clock Phase Shift Required (Late)
------------------------------------------------------------------
From            To      View        L->L     L->T   T->L     T->T
------------------------------------------------------------------
```

```
EJ_TCK           EJ_TCK setup_view1 17.400   0.000  17.400  17.400
virt_in_EJ_TCK   EJ_TCK setup_view1 17.400   0.000  17.400  17.400
-------------------------------------------------------------------
```

- The `report_clocks` command output displays a column showing active or inactive clocks in the design:

```
------------------------------------------------------------
Clock Descriptions
------------------------------------------------------------
Attributes
------------------------------------------------------------
Clock Name    Source   Period    Lead    Trail   Generated   Propagated   Active
-------------------------------------------------------------------------------
CLK_W_1       CLK1     20.000    0.000   10.000   n           y            y
CLK_W_2       CLK2     25.000    0.000   12.500   n           y            y
CLK_W_4_gen2  u8/Q     20.000    0.000   10.000   y           y            y
-------------------------------------------------------------------------------
```

Related Information

- create_clock
- create_generated_clock
- set_clock_uncertainty

report_constraint

```
report_constraint
[-all_violators]
[-late]
[-early]
[-check_type { pulse_width | clock_period | recovery | removal | clock_gating_setup |
clock_gating_hold | skew | pulse_clock_max_width |
pulse_clock_min_width | electromigration | rail_swing}]
[-drv_violation_type { max_capacitance | max_transition | max_fanout | min_capacitance |
min_transition | min_fanout | pulse_clock_max_transition | pulse_clock_min_transition}]
[-connection_class]
[-retime {aocv | path_slew_propagation | aocv_path_slew_propagation}]
[-retime_mode {path | exhaustive}]
[-view viewName]
[pin_port_list]
[-verbose]
[{> | >>} filename[.gz]]
```

Reports constraint information of current design. The constraints information includes information such as whether or not constraint is violated, by how much amount, and the worst violator object.

The following timing checks are reported when you use the `report_constraint` command:

- All setup and hold checks
- DRV
- Clock and asynchronous checks including pulse_width, min_period, recovery, removal, clock_gating_setup, clock_gating_hold, skew

The `report_constraint` command does not honor the `timing_report_fields` attribute by default. To honor this attribute you need to set the `timing_report_constraint_use_report_timing_format` attribute to `true`.

The `report_constraint` command supports DRV constraints (`set_min_fanout`, `set_max_fanout`, `set_min_transition`, `set_max_transition`, `set_min_capacitance`, and `set_max_capacitance`) applied on library pins.

The `report_constraint` command supports multi-threaded reporting. You can make the following settings to enable multi-threaded reporting:

```
set_multi_cpu_usage -localCpu number_of_CPU
set_table_style -no_frame_fix_width
```

By default, the `report_constraint` command reports only the mean slews. You can enable DRV

checking to use mean as well as the n-sigma slew values (in SOCV mode), with the following setting:
`set timing_report_max_transition_check_using_nsigma_slew true`

Parameters

`{> | >>}` *`filename`*	When specified with `>`, writes the report to the specified file name. If the file already exists, the software overwrites it. When specified with `>>`, writes the report to the specified file name. If the file already exists, the software concatenates the report to the end of the file. The *`filename`* parameter must be the last argument in the list. **Note:** To write a compressed report, add the `.gz` extension to the file name. *Default:* Report is displayed on standard output without being saved.

`-all_violators`	Reports negative slack values for timing checks. The supported check types are: • setup • hold • clock_gating_setup • clock_gating_hold • recovery • removal • clock_period • pulse_width • max_capacitance • max_transition • max_fanout • min_capacitance • min_transition • min_fanout • skew • pulse_clock_max_width • pulse_clock_min_width • pulse_clock_max_transition • pulse_clock_min_transition You can use the `-all_violators` parameter for reporting specific check types to generate report for that particular check type only. For example, you can use `-all_violators` with `-late` to report setup checks and path delays from set_max_delay.
`-check_type`	Reports setup or hold checks. The supported check types are: • clock_gating_setup: Reports only the paths that

end at the clock_gating_setup timing check.

- clock_gating_hold: Reports only the paths that end at the clock_gating_hold timing check.
- electromigration: Reports signal EM analysis related violations. When high density current passes through a thin metal wire, the high energy electrons exert forces on atoms causing them to migrate. Electromigration (EM) can drastically reduce the lifetime of a chip by increasing the resistivity of a metal wire or creating a short circuit between adjacent lines. When this option is specified, the software reports standard cell level electromigration violations.
- recovery: Reports paths that end at the recovery timing check.
- removal: Reports paths that end at the removal timing check.

  Note: The check types - recovery, removal, clock_gating_setup, and clock_gating_hold are also supported in AOCV mode.
- pulse_width: Reports only the paths that end at the pulse_width timing check.
  If you do not specify the `-all_violators` parameter, then the `pulse_width` option reports only one entry of the most critical pulse_width check. The DRV checks also report only the most critical entry.

  To report all the violated pulse_width checks, you need to specify `report_constraint -check_type pulse_width -all_violators` command (which is consistent with other check types of the `report_constraint` command).

	To report both violating and met entries, you can use the `report_min_pulse_width` command. • clock_period: Reports only the paths that end at the clock_period timing check. • skew: Reports skew checks. The `report_constraint` command performs skew checks by default. Skew checks are implemented as setup checks that are reported in the late analysis or simultaneous mode. These checks are not reported in the early mode. Skew checks are performed assuming the signal and reference clocks have the same frequency. If the frequencies of the signal and reference clocks are different, a warning message is displayed. • pulse_clock_max_width: Reports the maximum pulse clock width checks. • pulse_clock_min_width: Reports the minimum pulse clock width checks. • `rail_swing`: Reports details related to the rail swing checks. For rail swing check settings, you can use the following attributes: ◦ `timing_rail_swing_checks_low_voltage_threshold` ◦ `timing_rail_swing_checks_high_voltage_threshold`
`-connection_class`	Reports violations if the connection class attributes of the driving pin and receiving pin of a net are different.
`-drv_violation_type`	Reports DRV violations. The supported DRV check types are: • max_capacitance: Reports pins having the worst `max_capacitance` constraint. If you specify

the `-all_violators` parameter, you can see all the pins whose actual capacitance is greater than the `max_capacitance` value specified in library and timing constraints file.

- min_capacitance: Reports pins having the worst `min_capacitance` constraint. If you specify the `-all_violators` parameter, you can see all the pins whose actual capacitance is less than the `min_capacitance` value specified in the library and timing constraints file.
- max_transition: Reports pins having the worst `max_transition` constraint. If you specify the `-all_violators` parameter, you can see all the pins whose actual transition is greater than the `max_transition` value specified in the library and timing constraint file.

  Note: These checks are not reported in hold analysis mode. The `set_si_mode -enable_drv_with_delta_slew true` setting is required for max_transition DRV violations with SI slew.
- min_transition: Reports pins having the worst `min_transition` constraint. If you specify the `-all_violators` parameter, you can see all the pins whose actual transition is less than the `min_transition` value specified in the library and timing constraints file.
- max_fanout: Reports pins having the worst `max_fanout` constraint. If you specify the `-all_violators` parameter, you can see all the pins whose actual fanout is greater than the `max_fanout` value specified in the library and timing constraints file.
- min_fanout: Reports pins having the worst `min_fanout` constraint. If you specify the `-`

	`all_violators` parameter, you can see all the pins whose actual fanout is less than the `min_fanout` value specified in the library and timing constraints file. • pulse_clock_max_transition: Reports maximum pulse clock transition checks. • pulse_clock_min_transition: Reports minimum pulse clock transition checks.
`-early`	Reports all of the hold checks, including regular, clock_gating_hold, and data-to-data hold checks. Also reports path delays from set_min_delay. **Note:** This parameter does not report removal checks.
`-late`	Reports all of the setup checks, including regular, clock_gating_setup, and data-to-data setup checks. Also reports path delays from set_max_delay. **Note:** This parameter does not report recovery checks.
`pin_port_list`	Controls the output of DRV constraints. When specified, all DRV violations are reported for the specified pins or ports only.

`-retime {aocv | path_slew_propagation | aocv_path_slew_propagation}`	Reanalyzes the specified set of paths using the specified method. **Note**: The parameter supports setup/hold check types only. • `aocv`: Re-evaluates the given set of graph-based analysis (GBA) paths by reverting back graph-based AOCV derate factors and applying path- based AOCV derates instead of less pessimistic derates. In this case the slew (and respective delays) values are worst that are propagated during graph-based analysis. • `aocv_path_slew_propagation`: Re-evaluates the given set of graph-based analysis (GBA) paths by using path-based AOCV derates and delays using actual path- based propagated slew. • `path_slew_propagation`: Re-evaluates the given set of graph-based analysis (GBA) paths by recalculating the delay values based upon actual propagated slew across the path. In this case the AOCV derates are the ones that are reported during graph-based analysis.

`-retime_mode {path | exhaustive}`	Allows the `report_constraint` command to be run in exhaustive path-based analysis (PBA) mode. In this mode, the software will exhaustively examine nworst paths of the selected endpoints and then select the true worst path to be reported - based on retimed slacks of all the examined paths. To enable exhaustive mode, the use model is given below: `report_constraint -all_violators -retime path_slew_propagation -retime_mode exhaustive` When you use this parameter, reporting can be run time intensive as the software needs to do an exhaustive search for true worst paths. Therefore, it is recommended to use this option in the later phases of sign-off stage when there are very few violations left for analysis. *Default*: `path`
`-verbose`	Generates a report that displays the full data path with the accompanying required time and slack calculation.
`-view` *viewName*	Generates a report containing endpoint and other constraint information for the specified analysis view. You can only specify this parameter when the software is in multi-mode multi-corner analysis mode. *Default:* Generates a report containing the worst endpoint and constraint information based on all of the analysis views.

Examples

- Consider a design having 3 sequential endpoints:
- setup_pin: Having a setup check with slack equal to -9.000

- recovery_pin: Having a recovery check with slack equal to -8.000
- both_setup_recovery_pin: Having both setup check (slack = -6.000) and recovery check (slack = -7.000)
  In this case, the report_constraint -late -all_violators command will display the following output:

```
max_delay/setup
-------------------------------------------------------
End Point                  Slack          Cause
-------------------------------------------------------
setup_pin f                -9.000         VIOLATED
both_setup_recovery_pin    -6.000         VIOLATED
-------------------------------------------------------
```

- The `report_constraint -check_type recovery -all_violators` command will display the following output:

```
Check type : recovery
-------------------------------------------------------------
End Point                     Slack          Cause
-------------------------------------------------------------
recovery_pin r                -8.000         VIOLATED
both_setup_recovery_pin       -7.000         VIOLATED
-------------------------------------------------------------
```

- The `report_constraint -late -check_type recovery -all_violators` command displays the following output:

```
max_delay/setup
-----------------------------------------------------------------------------
End Point                  Slack           Cause
-----------------------------------------------------------------------------
setup_pin f                -9.000          VIOLATED
recovery_pin r             -8.000          VIOLATED
both_setup_recovery_pin    -7.000          VIOLATED
-------------------------------------------------------------
```

- The `report_constraint` command displays the following output:

```
max_delay/setup
```

```
------------------------------------------------------------
End Point            Slack                Cause
------------------------------------------------------------
out2 r              -4111.834             VIOLATED
------------------------------------------------------------
Check type : clock_period
---------------------------
No paths found
Check type : pulse_width
------------------------------------------------------------------------
                         SEQUENTIAL CLOCK PULSE WIDTH
------------------------------------------------------------------------
Pin             Required         Actual Pulse         Slack
                Pulse Width      Width
------------------------------------------------------------------------
dff2/CK (low)   0.260            -670.800            -671.060
------------------------------------------------------------------------
Check type : max_transition
---------------------------
Pin : sub1/dff1/D
max_transition : 4.500
Transition Time: 5114.060
--------------------------------------------------------------------------------
slack: -5109.560 (VIOLATED)
Check type : max_capacitance
---------------------------
Pin : buf133/Y
max_capacitance: 0.000
-Capacitance : 0.423
---------------------------
slack : -0.423 (VIOLATED)
Check type : max_fanout
---------------------------
Pin : specialClkGate/o
max_fanout : 0.557
```

```
fanout : 1.000
---------------------------
slack : -0.443 ( VIOLATED )
```

- The `report_constraint -all_violators` command displays the following output:

```
max_delay/setup

-----------------------------------------------------------
End Point                Slack               Cause
-----------------------------------------------------------
out2 r                   -4111.834           VIOLATED
dff3/D r                 -4110.260           VIOLATED
lat2/D r                 -3774.174           VIOLATED
dff2/D r                 -3770.285           VIOLATED
out r                    -3161.115           VIOLATED
and1/B r                 -2464.880           VIOLATED
dff1/D r                 -2098.868           VIOLATED
sub2/bufH1/dff1/D r      -2098.821           VIOLATED
sub2/bufH2/dff1/D r      -2028.843           VIOLATED
sub1/dff1/D r            -2028.843           VIOLATED
dff5/D r                 -1279.155           VIOLATED
-----------------------------------------------------------
Check type : clock_period
---------------------------
No paths found
Check type : skew
---------------------------
No paths found
Check type : pulse_width
-------------------------------------------------------------------
                            SEQUENTIAL CLOCK PULSE WIDTH
-------------------------------------------------------------------
Pin                        Required      Actual Pulse     Slack
                           Pulse Width   Width
-------------------------------------------------------------------
```

```
dff2/CK (low)                0.260       -670.800        -671.060
lat2/GN (low)                0.201       -670.800        -671.001
sub2/bufH1/dff1/CK (high)    0.167       -670.800        -670.968
dff1/CK (high)               0.167       -670.800        -670.968
sub2/bufH2/dff1/CK (high)    0.167       -670.800        -670.968
sub1/dff1/CK (high)          0.167       -670.800        -670.968
dff11/CK (high)              0.167       -670.800        -670.968
-------------------------------------------------------------------
Check type : max_transition
-------------------------------------------------------------------
Pin Name                 Required       Actual         Slack
-------------------------------------------------------------------
and_parallel_1/A         4.500          5114.060      -5109.560
buf4/A                   4.500          5114.060      -5109.560
sub1/dff1/D              4.500          5114.060      -5109.560
Check type : max_capacitance
-------------------------------------------------------------
Pin Name                  Required        Actual      Slack
-------------------------------------------------------------
buf133/Y                  0.000           0.423       -0.423
buf11/Y                   0.000           0.105       -0.105
buf5/Y                    0.000           0.105       -0.105
dff11/Q                   0.000           0.105       -0.105
Check type : max_fanout
-------------------------------------------------------------
Pin Name                  Required        Actual      Slack
-------------------------------------------------------------
specialClkGate/o          0.557           1.000       -0.443
```

- The `report_constraint -all_violators -drv_violation_type min_capacitance` command reports the following:

```
-----------------------------------------------
Pin Name    Required     Actual       Slack
-----------------------------------------------
DIN         0.0400        0.0013      -0.0387
```

```
F2/Q        0.0400        0.0013       -0.0387
CLK1        0.0400        0.0051       -0.0349
```

- The `report_constraint -drv_violation_type min_capacitance` command outputs the following:

```
Check type : min_capacitance
---------------------------
Pin : F2/Q
Capacitance: 0.0013
- min_capacitance: 0.0400
-----------------------------------------------------
slack: -0.0387 ( VIOLATED )
```

- The `report_constraint -all_violators -drv_violation_type min_transition` command outputs the following:

```
-----------------------------------------------
Pin Name    Required      Actual      Slack
-----------------------------------------------
I0/A        0.0300        0.0001       -0.0299
I1/A        0.0300        0.0001       -0.0299
CLK0        0.0300        0.0001       -0.0299
CLK1        0.0300        0.0001       -0.0299
DIN         0.0300        0.0001       -0.0299
I4/S0       0.0300        0.0001       -0.0299
F1/DATA     0.0300        0.0001       -0.0299
SEL         0.0300        0.0001       -0.0299
```

- The `report_constraint -drv_violation_type min_transition` command outputs the following:

```
Check type : min_transition
---------------------------
Pin : F1/DATA


Transition Time:0.0001
- min_transition :0.0300
-----------------------------------------------------
slack : -0.0299 ( VIOLATED )
```

- The `report_constraint -all_violators -drv_violation_type min_fanout` command outputs the following:

```
-----------------------------------------------------
Pin Name    Required       Actual        Slack
-----------------------------------------------------
F5/Q       2.0000         0.0000         -2.0000
CLK0       2.0000         1.0000         -1.0000
F3/Q       2.0000         1.0000         -1.0000
F2/Q       2.0000         1.0000         -1.0000
F1/Q       2.0000         1.0000         -1.0000
SEL        2.0000         1.0000         -1.0000
CLK1       2.0000         1.0000         -1.0000
```

- The `report_constraint -drv_violation_type min_fanout` command outputs the following:

```
Check type : min_fanout
---------------------------
Pin : F5/Q
Fanout: 0.0000
- min_fanout: 2.0000
-----------------------------------------------------
slack : -2.0000 ( VIOLATED )
```

- The `report_constraints -check_type clock_period` command reports the following:

```
Check type : clock_period
-----------------------------------------------------------
End Point                Slack      Cause
-----------------------------------------------------------
ffo_ram/mm0/mm1/CLKB f   -0.027     VIOLATED
-----------------------------------------------------------
```

Related Commands

- set_min_fanout
- set_min_fanout
- set_min_transition
- set_max_transition
- set_min_capacitance
- set_max_capacitance
- report_timing

report_cppr

```
report_cppr
[-help]
[-check_type {setup | hold | clock_gating_setup | clock_gating_hold |
clock_gating_pulse_width | data_setup | data_hold | recovery | removal | pulse_width |
clock_period | clock_separation | skew | no_change_setup | no_change_hold | time_borrow}]
-from pin_or_port
-to pin_or_port
[-from_clock clkname]
[-to_clock clkname]
[-early | -late]
[-view view_name]
[> filename]
```

Reports the clock reconvergence pessimism value at the common branching point of the early and late paths in the clock network of the specified data path.

The `report_cppr` command reports separate values for both opening and closing edges for paths with latches as destinations. If both source and destination flops are triggered by multiple clocks, then one report is generated for each separate clock.

When the software is in multi-mode multi-corner analysis mode, `report_cppr` reports the results for each active analysis view.

To use the `report_cppr` command, you must first set the following attributes:

- `timing_cppr_transition_sense`
- `timing_cppr_threshold_ps`

Parameters

`-help`	Prints out the command usage.

`-check_type {setup | hold | clock_gating_setup | clock_gating_hold | clock_gating_pulse_width | data_setup | data_hold | recovery | removal | pulse_width | clock_period | clock_separation | skew | no_change_setup | no_change_hold | time_borrow}`	Reports the CPPR for the paths that end at the specified timing check. The following check types are reported when the software is in setup analysis mode: `setup`, `pulse_width`, `clock_period`, `clock_gating_setup`, `clock_gating_pulse_width`, `data_setup`, `recovery`, skew, `clock_separation`, `time_borrow`, and `no_change_setup`. The following check types are reported when the software is in hold analysis mode: `hold`, `clock_gating_hold`, `data_hold`, `removal`, and `no_change_hold`. If the software is in simultaneous setup and hold analysis mode, all of the check types listed above are reported.
`> filename`	Writes the report to the specified file name. If the file already exists, the software overwrites it. The *`filename`* parameter must be the last argument in the list. **Note:** To write a compressed report, add the `.gz` extension to the file name. *Default:* Report is displayed on standard output without being saved.
`-early`	Reports the hold analysis clock reconvergence pessimism value. You can specify this parameter when the software is in hold analysis mode, or in simultaneous setup and hold analysis mode (`set_db timing_enable_simultaneous_setup_hold_mode true`).
`-from` *`pin_or_port`*	Reports the clock reconvergence pessimism value for paths starting from the specified clock pin of the source flop, or from the specified port.

`-from_clock` *clkname*	Reports the clock reconvergence pessimism value for paths with the specified triggering clock. Use this parameter when there is more than one clock triggering the source flop.
`-late`	Reports the setup analysis clock reconvergence pessimism value. You can specify this parameter when the software is in setup analysis mode, or in simultaneous setup and hold analysis mode (`set_db timing_enable_simultaneous_setup_hold_mode true`).
`-to` *pin_or_port*	Reports the clock reconvergence pessimism value for paths ending at the specified clock pin of the destination flop, or at the specified port.
`-to_clock` *clkname*	Reports the clock reconvergence pessimism value for paths with the specified capturing clock. Use this parameter when there is more than one clock capturing the destination flop.
`-view` *view_name*	Reports the clock reconvergence pessimism values for paths in the specified analysis view only. You can specify this parameter only when the software is in multi-mode multi-corner timing analysis mode.

Examples

Here are some examples.

- When the software is in multi-mode multi-corner timing analysis mode, you can specify the `-view` parameter to report the CPPR values for a specific analysis view:

```
report_cppr -from CLK3 -to seg3/u14/CK


Start Clock Pin: CLK3 (Triggered by rising edge of clock CLK_W_3)
End Clock Pin: seg3/u14/CK (Triggered by rising edge of clock CLK_W_3)
Analysis Type: Setup
Analysis View: setup_view1
```

```
Common Branch Point : seg3/u17_c/Y
Launching Edge Type At Common Point : Rising
Capturing Edge Type At Common Point : Rising
Pessimism Threshold Value : 0.020
-------------------------------------
Arrival   Late    Early   Pessimism
Times
-------------------------------------
Rise      0.549   0.366   0.183
Fall      0.400   0.267   0.133
-------------------------------------
Edge Selection Mode : Normal
Edge Selection : Match, Using Rise
CPPR Adjustment : 0.183
```

- The following report shows sigma and mean values:

```
report_cppr -from FF01/CP -to FF02/CP


Start Clock Pin: FF01/CP (Triggered by rising edge of clock CK)
End Clock Pin: FF02/CP (Triggered by rising edge of clock CK)
Analysis Type: Setup
Analysis View: view1
Common Branch Point : CK
Launching Edge Type At Common Point : Rising
Capturing Edge Type At Common Point : Rising
Pessimism Threshold Value : 0.020
+-------------------------------------------------------------------------------+
| Arrival | Late  | Late  | Early | Early | Pessimism | Pessimism | Pessimism |
| Times   | Mean  | Sigma | Mean  | Sigma | Mean      | Sigma     |           |
|---------+-------+-------+-------+-------+-----------+-----------+-----------|
| Rise    | 0.000 | 0.000 | 0.000 | 0.000 | 0.000     | 0.000     | 0.000     |
| Fall    | 0.000 | 0.000 | 0.000 | 0.000 | 0.000     | 0.000     | 0.000     |
+-------------------------------------------------------------------------------+
Edge Selection Mode : Normal
Edge Selection : Match, Using Rise
CPPR Adjustment : 0.000 (+3.00S) | 0.000 | 0.00
```

report_design

```
report_design
[-view viewName]
[{> | >>} filename | -tcl_list]
[-early | -late]
```

Reports the minimum and/or maximum operating conditions and active design rules for the current design.

Both minimum and maximum operating conditions are available for on-chip variation analysis mode.

When simultaneous setup hold analysis mode is set to `on`, the minimum or maximum conditions can be reported using the `-early`/`-late` parameters.

Parameters

{> | >>} *filename*	When specified with >, writes the report to the specified file name. If the file already exists, the software overwrites it. When specified with >>, writes the report to the specified file name. If the file already exists, the software concatenates the report to the end of the file. The *filename* parameter must be the last argument in the list. The *filename* and -tcl_list parameters are mutually exclusive; you cannot specify them together. **Note:** To write a compressed report, add the .gz extension to the file name. *Default:* Report is displayed on standard output without being saved.
-early | -late	The -early parameter reports minimum operating condition in bcwc or onChipVariation analysis mode. The -late parameter reports maximum operating condition. The -late and -early parameters are mutually exclusive.
-tcl_list	Produces the report in Tcl list format instead of a tabular format. This is useful for integrating timing with custom Tcl functions, and also for customizing report generation. The -tcl_list and *filename* parameters are mutually exclusive; you cannot specify them together.
-view *viewName*	Reports view specific operating conditions.

Examples

- The following is a sample report generated using the report_design command:

```
report_design
```

Design Property	Value
Design Name	dsp_core
Operating Condition Name	slow
Process	1.000
Voltage	1.620
Temperature	125.000
Tree Type	balanced

Design Rule	Value
Max Transition	NA
Min Transition	NA
Max Capacitance	NA
Min Capacitance	NA
Max Fanout	24.000

- The following sample report shows both the minimum and maximum operating condition details in simultaneous mode:

```
report_design
```

Design Property	Value	
Design Name	dsp_test	
	MAX	**MIN**
Operating Condition Name	tdsp_core/%NOM_PVT	tdsp_core/%NOM_PVT
Process	1.000	1.000
Voltage	1.800	1.800

Temperature	25.000	25.000
Tree Type	worst_case	worst_case

Design Rule	Value
Max Transition	NA
Min Transition	NA
Max Capacitance	NA
Min Capacitance	NA
Max Fanout	NA
Min Fanout	NA

Related Commands

- report_timing
- write_db

report_fanin

```
report_fanin
-to pin_port_net_list
[-trace_through {case_disable | user_disable | all}]
[-pin_levels numOfPinLevels]
[-view viewName]
[-nosplit]
[{> | >> } filename]
```

Allows a cone traversal that is not tied to the timing graph, that is, not blocked by `set_disable_timing` and case analysis. By default, the trace matches the timing view and the reports generated using the `all_fanin` and `all_fanout` commands. The reports indicate whether disabled or case analysis arcs were traversed.

Parameters

`> | >> filename`	When specified with >, writes the report to the specified file name. If the file already exists, the software overwrites it. When specified with >>, writes the report to the specified file name. If the file already exists, the software concatenates the report to the end of the file.
`-nosplit`	Specifies that reports with long names will not be split into multiple lines.
`-pin_levels numOfPinLevels`	Specifies the desired depth of the output.

`-trace_through {case_disable | user_disable | all}`	Allows traversal through disabled arcs. You can specify one of the following options: • `case_disable`: Arcs which are disabled because of set_case_analysis are traversed but because of set_disable_timing are not traversed • `user_disable`: Arcs disabled because of set_disable_timing are traversed but because of set_case_analysis are not traversed • `all`: Trace through all arcs
`-to` *pin_port_net_list*	Specifies a list or collection of pins, ports, or nets. A mixed list or heterogeneous collection is also allowed.
`-view` *viewName*	Specifies disabled arcs in the specified view.

Examples

- The `report_fanin -to h0/A` command shows the following output:

```
Fanin Network for Pin: h0/A
-----------------------------------------------------------------------
Source Pin   Sink Pin  Cell Type   Timing Sense    Arc       Case
                                                   Disabled  Analysis
-----------------------------------------------------------------------
u1/Y         h0/A      (net)       -                -         -
-----------------------------------------------------------------------
u1/A         u1/Y      BUFX2       positive_unate   -         -
-----------------------------------------------------------------------
in           u1/A      (net)       -                -         -
-----------------------------------------------------------------------
```

Related Commands

- set_disable_timing
- set_case_analysis
- all_fanin
- all_fanout

report_fanout

```
report_fanout
{-from pin_port_net_list | -clock_tree}
[-trace_through {case_disable | user_disable | all}]
[-pin_levels numOfPinLevels]
[-view viewName]
[-nosplit]
[{> | >> } filename]
```

Allows a cone traversal that is not tied to the timing graph, that is, not blocked by `set_disable_timing` and case analysis. By default, the trace matches the timing view and the reports generated using the `all_fanin` and `all_fanout` commands. The reports indicate whether disabled or case analysis arcs were traversed.

Parameters

`> | >> filename`	When specified with >, writes the report to the specified file name. If the file already exists, the software overwrites it. When specified with >>, writes the report to the specified file name. If the file already exists, the software concatenates the report to the end of the file.
`-clock_tree`	Reports the fanout of the source pin, one after the other for all the clocks in the design.
`-nosplit`	Specifies that reports with long names will not be split into multiple lines.
`-pin_levels numOfPinLevels`	Specifies the desired depth of the output.

`-trace_through {case_disable | user_disable | all}`	Allows traversal through disabled arcs. You can specify one of the following options: • `case_disable`: Arcs which are disabled because of set_case_analysis are traversed but because of set_disable_timing are not traversed • `user_disable`: Arcs disabled because of set_disable_timing are traversed but because of set_case_analysis are not traversed • `all`: Trace through all arcs
`-from` *`pin_port_net_list`*	Specifies a list or collection of pins, ports, or nets. A mixed list or heterogeneous collection is also allowed.
`-view` *`viewName`*	Specifies disabled arcs in the specified view.

Examples

- Suppose arc seg1/u1 is disabled due to following constraint:
  `set_disable_timing -from A -to Y seg1/u1`

  The report_fanout `-from IN1` command will show the following output:

```
Fanout Network for Port: IN1
------------------------------------------------------------------------
Source Pin  Sink Pin    Cell Type   Timing Sense   Arc       Case
                                                   Disabled  Analysis
------------------------------------------------------------------------
IN1         seg1/u1/A   (net)       -              -         -
------------------------------------------------------------------------
```

- The r`eport_fanout -from IN1 -trace_through user_disable` command will show the following output:

```
Fanout Network for Port: IN1
------------------------------------------------------------------------
Source Pin    Sink Pin   Cell Type  Timing Sense    Arc       Case
                                                    Disabled  Analysis
------------------------------------------------------------------------
IN1           seg1/u1/A  (net)      -               -         -
------------------------------------------------------------------------
seg1/u1/A     seg1/u1/Y  CLKBUF     positive_unate  Y         -
------------------------------------------------------------------------
seg1/u1/Y     seg1/u2/A  (net)      -               -         -
------------------------------------------------------------------------
```

Related Commands

- set_disable_timing
- set_case_analysis
- all_fanin
- all_fanout

report_inactive_arcs

```
report_inactive_arcs
[-help]
[instance_port_obj]
[-exclude_type {const snipped disable library_disable global_disable conditional_disable
library_mode}]
[-include_net_arcs]
[-include_when_cond]
[-type {const snipped disable library_disable global_disable conditional_disable
library_mode power_mode_disable}]
[-view view_name]
[> <filename[.gz]>]
[>> <filename[.gz]>]
[-delay_arcs_only | -check_arcs_only]
```

Reports information about disabled timing and timing check arcs in the design. Reports all arcs disabled due to user-specified exceptions such as `set_disable_timing` or `set_case_analysis`, as well as information about arcs disabled by constant propagation during timing analysis, snipped loop arcs, and arcs disabled in the timing library.

Parameters

`-help`	Prints out the command usage.
instance_port_obj	Reports disabled timing arcs for the specified list of instances or ports. If you specify a list of instances, the software reports all disabled arcs of the specified cells. If you specify a list of ports, the software reports all disabled arcs directly connected to the ports. *Default:* Reports all disabled net and cell arcs.
`-check_arcs_only`	Lists only disabled check arcs. The `check_arcs` are disabled by using the `set_disable_timing` command. *Default*: Both disabled timing delay arcs and check arcs are listed.

`-delay_arcs_only`	Lists only disabled timing delay arcs.
`-exclude_type {const snipped disable library_disable global_disable conditional_disable library_mode}`	Excludes the disabled arcs that belong to the specified category of disabled arcs. The valid options are: const, snipped, disable, library_disable, global_disable, conditional_disable, and, library_mode.
`-include_net_arcs`	Includes disabled net arcs in the report.
`-include_when_cond`	Shows the when_cond of arcs in the output report.
`-type {const snipped disable library_disable global_disable conditional_disable library_mode power_mode_disable}`	Lists only the disabled arcs that belong to the specified category of disabled arcs. The `-type` options restrict the report to only those arcs that are disabled due to a particular timing analysis step, such as constant propagation, specified disables, or loop arc disables. The valid options are: • `const`: Propagated constant -- arcs disabled due to a constant value. The constant value can apply to one of the arc nets/pins, to side inputs which disable a condition for the arc, to nets that disable an active mode for the arc, or disable a default conditional arc. • `snipped`: Loop snipped arcs -- arcs automatically disabled by the software as part of a feedback loop. • `disable`: Defined by using the `set_disable_timing` command. • `library_disable`: Delay arc disabled due to library disables. These disables may be caused by the following conditions: ◦ Disables specified using the `set_disable_timing` command. ◦ The arc is an asynchronous arc and the

	attribute `timing_enable_preset_clear_arcs` is set to `false`. • `global_disable`: Lists the arcs that are disabled through user-specified global variable settings. • `conditional_disable`: Arcs disabled if a `when` condition is specified. There is no constant on sink or source pin and arc is disabled due to the `when` condition. • `library_mode`: Displays library mode of disabled arcs. • `power_mode_disable`: Reports the arcs for which power mode is disabled.
`-view` *view_name*	Generates the report for the specified analysis view only. You can specify this parameter only when the software is in multi-mode multi-corner timing analysis mode.
`{> | >>}` *filename*	When specified with `>`, writes the report to the specified file name. If the file already exists, the software overwrites it. When specified with `>>`, writes the report to the specified file name. If the file already exists, the software concatenates the report to the end of the file. The *filename* parameter must be the last argument in the list. **Note:** To write a compressed report, add the `.gz` extension to the file name. *Default:* Report is displayed on standard output without being saved.

Examples

- The following command reports all disabled arcs in the design:

  ```
  report_inactive_arcs > Report
  ```

  The following example shows the report generated:

  ```
  ------------------------------------------------
  Flags :   const                propagated constant
            snipped              loop snipped arcs
            disable              set_disable_timing
            disable_clock_gating set_disable_clock_gating
            library_disable      set_disable_cell_timing
            Missing_Phase        No Arc Phase Data
  From         To           DisableType ArcType              Reason
  ------------------------------------------------------------------------
  u8/CK ^      u8/D ^       const       hold_rising_clk_rise  u8/D = 1
  u8/CK ^      u8/D v       const       hold_falling_clk_rise u8/D = 1
  seg3/u12/A ^ seg3/u12/Y v disable     combinational         User Disable
  ------------------------------------------------------------------------
  ```

- When the software is in multi-mode multi-corner analysis mode, the report generated lists the disabled arcs by active analysis view, as shown in the following example:

  ```
  ------------------------------------------------
  Flags : const propagated constant
  snipped loop snipped arcs
  disable set_disable_timing
  disable_clock_gating set_disable_clock_gating
  library_disable set_disable_cell_timing
  Missing_Phase No Arc Phase Data
  From        To                 DisableType ArcType  Reason         View
  ------------------------------------------------------------------------------

  gscanmode ^ FE_O305_gmode/I ^  const       net_arc  gscanmode = 0  setup_view2
  gscanmode v FE_O305_gmode/I v  const       net_arc  gscanmode = 0  setup_view2
  gscanmode ^ FE_O305_gmode/I ^  const       net_arc  gscanmode = 0  hold_view1
  gscanmode v FE_O305_gmode/I v  const       net_arc  gscanmode = 0  hold_view1
  gscanmode ^ FE_O305_gnmode/I ^ const       net_arc  gscanmode = 0  hold_view2
  ------------------------------------------------------------------------------
  ```

Related Commands

- set_disable_timing
- set_case_analysis

report_instance_library

```
report_instance_library
-out_file fileName
[-instance instanceName]
[-view viewName | -delay_corner delayCornerName]
[-power_domain domainName]
[-early | -late]
```

Reports library related information for an instance(s). The report prints the following information about an instance:

- Instance name
- Cell type
- View name or delay corner
- Power domain name
- Library/Libset (early/late)
- Library file (early/late)
- Operating conditions
- Binding Status : Prints out the status for the matched voltage library available. If the user setting is `True`, then the binding status is not displayed.

Parameters

`-delay_corner` *delayCornerName*	Specifies the delay corner from which an instance library is to be selected. If no delay corner is specified, then all the instance libraries will be printed.
`-early` | `-late`	Specifies the library set (early/late) from which a library instance is to be printed. If either of these options are not specified then both the early and late library sets are used.

`-out_file` *fileName*	Specifies the name of the output file. If no file is specified, the report will be printed on standard output file.
`-instance` *instanceName*	Specifies the instance(s) for which the report is to be printed. If no instance is provided the report will print information of all the instances. The value of instance can be a list of regular expressions.
`-power_domain` *domainName*	Specifies the power domain from which instance(s) are to be selected. This option should be specified with the `-delay_corner` or `-view` options.
`-view` *viewName*	Specifies the view from which instance library is to be selected.

Examples

- The `report_instance_library` command displays the following information about instance `ff2` :

```
Instance              : ff2
Analysis View         : view1
Power Domain          : Default
Library/Libset(early) : locv/libCom
Library File (early)  : locv/locv2.lib
Library/Libset(late)  : locv/libCom
Library File (late)   : locv/locv2.lib
Op Cond               : P->1.000000, V->3.300000, T->25.000000 (locv/%NOM_PVT)
Cell Type             : gateCell
```

- The following command displays the following information about instance AO1/g1080:

```
report_instance_library -instance AO1/g1080

Instance : AO1/g1080
Cell : INVXL
Analysis View : AV_bc_on
Power Domain : Default
Library/Libset : slow/INV.lib
Library File : INV.lib
User Setting : True
Op Cond : P->1.000000, V->1.500000, T->0.000000 (fast)
Cell Type : gateCell
```

Related Commands

- check_timing_library_consistency

report_min_pulse_width

```
report_min_pulse_width
[-help]
[-clock <object_list>]
[-net]
[-path_type {summary | short | full_clock}]
[-pins object_list]
[-tcl_list]
[-verbose]
[-net]
[-violation_only]
[-view view_name]
[-waveform_aware_type {regular | waveform_aware | both}]
[ { > | >> } filename]
```

Reports a set of minimum pulse width violations on pins and ports on the clock network. The report shows the required pulse width, the actual pulse width, and by how much the constraint is violated or met. This is reported separately for sequential elements and instances on the clock tree.

You can customize pulse width reporting format by using the `timing_report_fields` attribute.

The report header shows clock edge information to make it more intuitive for clock style checks.

Parameters

{> | >>} *filename*	When specified with >, writes the report to the specified file name. If the file already exists, the software overwrites it. When specified with >>, writes the report to the specified file name. If the file already exists, the software concatenates the report to the end of the file. The *filename* parameter must be the last argument in the list. The *filename* and -tcl_list parameters are mutually exclusive; you cannot specify them together. **Note:** To write a compressed report, add the .gz extension to the file name. *Default:* Report is displayed on standard output without being saved.
-clock <object_list>	Specifies a set of clocks to for which to report the minimum pulse width.
-net	Reports the net arcs. This parameter can be specified with the -path_type full_clock option only.

`-path_type {summary | short | full_clock}`	Specifies the type of report to generate. *Default:* `summary` • `summary`: Generates a summary report for each pulse width check that shows the required width, the actual width, and the resulting slack. • `short`: Generates a path report for each pulse width check that lists the latency of each edge of the pulse as a single lumped value, instead of reporting a detailed path report. • `full_clock`: Generates a path report for each pulse width check that shows the detailed pin-to-pin path for each edge of the pulse. **Note:** The `-path_type full_clock` parameter may cause an increased runtime. In that case you can use the `-path_type short` parameter.
`-pins` *`object_list`*	Specifies a set of ports and pins on the clock network. If this list is not specified, then the report contains all the pins and ports in the current design.
`-tcl_list`	Produces the report in Tcl list format instead of a tabular format. This is useful for integrating timing with custom Tcl functions, and also for customizing report generation. The `-tcl_list` and *`filename`* parameters are mutually exclusive; you cannot specify them together.
`-verbose`	Provides detailed information about minimum pulse width calculations. **Note:** This parameter is obsolete. The parameter still works in this release, but to ensure compatibility with future releases, use the `-path_type` parameter instead.

`-view` *view_name*	Generates the report for the specified analysis view only. You can specify this parameter only when the software is in multi-mode multi-corner timing analysis mode.
`-violation_only`	Reports only the negative slack minimum pulse width violations.
`-waveform_aware_type {regular | waveform_aware | both}`	Reports the waveform-aware pulse width checks. You can specify the following options: • `waveform_aware`: Reports only the waveform-aware pulse width checks. • `both`: Reports waveform-aware as well as regular pulse width checks. • `regular`: Reports regular pulse width checks. This is the default setting.

Examples

- The `report_min_pulse_width` will generate a summary report as shown below:

```
report_min_pulse_width

--------------------------------------------------
SEQUENTIAL CLOCK PULSE WIDTH
--------------------------------------------------
Pin                Required     Actual Pulse   Slack
                   Pulse Width  Width
--------------------------------------------------
seg3/u14/CK (high) 0.169        0.860         0.691
seg3/u3/CK (high)  0.169        0.875         0.706
seg3/u9/CK (high)  0.169        0.875         0.706
seg3/u3/CK (low)   0.195        1.083         0.888
seg3/u9/CK (low)   0.195        1.083         0.888
--------------------------------------------------
```

- A short report can be generated at any clock pin having pulse width check, as shown below. The pulse width is shown as sequential clock pulse width in case it is reported on sequential pins:

```
report_min_pulse_width -path_type short

SEQUENTIAL CLOCK PULSE WIDTH
```

```
----------------------------
Path 1: MET PulseWidth Check with Pin seg3/u14/CK
Ending Clock Edge: seg3/u14/CK (v) checked with trailing edge of 'CLK_W_3'
Beginning Clock Edge: seg3/u14/CK (^) triggered by leading edge of 'CLK_W_3'
Other End Arrival Time 1.503
- PulseWidth 0.169
+ Phase Shift 0.000
= Required Time 1.334
- Arrival Time 0.621
= Slack Time 0.714
Clock Rise Edge 0.000
+ Clock Network Latency (Prop) 0.621
= Beginpoint Arrival Time 0.621
----------------------------------------
Pin         Cell     Delay  Arrival  Edge
                            Time
----------------------------------------
seg3/u14/CK DFF              0.621   ^
----------------------------------------
Path 2: MET PulseWidth Check with Pin seg3/u3/CK
Ending Clock Edge: seg3/u3/CK (v) checked with trailing edge of 'CLK_W_3'
Beginning Clock Edge: seg3/u3/CK (^) triggered by leading edge of 'CLK_W_3'
Other End Arrival Time 1.578
- PulseWidth 0.169
+ Phase Shift 0.000
= Required Time 1.409
- Arrival Time 0.682
= Slack Time 0.727
Clock Rise Edge 0.000
+ Clock Network Latency (Prop) 0.682
= Beginpoint Arrival Time 0.682
---------------------------------------
Pin        Cell    Delay  Arrival   Edge
                          Time
---------------------------------------
seg3/u3/CK DFF             0.682     ^
---------------------------------------
```

Related Commands

- report_analysis_coverage
- report_timing
- set_min_pulse_width
- reset_min_pulse_width

report_mode

```
report_mode
inst_list
[-view viewName]
```

Reports the status of library modes associated with a specified instance. The report includes the status of each mode and its condition, for each mode group defined in the timing library.

Parameters

inst_list	Generates a report for the specified instances.
-view viewName	Reports the status of library modes in the specified active analysis view. Default: Reports the status of library modes in all active analysis views.

Examples

- The following command activates the `tst_mode_disable` library mode for instances of `i_ram`:
  ```
  set_mode tst_mode_disable [get_cells i_ram]
  ```

- The following command sets the active setup and hold analysis views:
  set_analysis_views -setup {v1 v2} -hold {v3}

- The following command reports the status of all modes associated with m4k_top:
  ```
  report_mode m4k_top

  ------------------------------------------------
  Mode Group tets_mode of m4k_top
  ------------------------------------------------
  Mode Name         Status    Condition     View Name
  ------------------------------------------------
  test_mode_enable  ACTIVE    TST_BYPASS    View1
  test_mode_disable INACTIVE  !(TST_BYPASS) View2
  ------------------------------------------------
  ```

- The following command reports the status of all modes in the analysis view `View1`:
  ```
  report_mode m4k_top -view View1

  ------------------------------------------
  Mode Group tets_mode of m4k_top
  ------------------------------------------
  Mode Name        Status  Condition  View Name
  ------------------------------------------
  test_mode_enable ACTIVE  TST_BYPASS View1
  ------------------------------------------
  ```

Related Commands

- set_analysis_view
- set_mode

report_nets

```
report_nets
[-min_fanout int]
[-max_fanout int]
{-net list_of_net_name_or_id | -pin list_of_pin_name_or_id}
[-tcl_list]
[-hier]
[[{> | >>} filename] | -tcl_list]
```

Reports the net information on the current module. The information includes the net name, the number of source pins, sink pins and bidirectional pins on the net, wire capacitance, and total capacitance.

Electrical information such as pin capacitance, and wire capacitance with which the net is associated are also included.

Use the `-pin` option to report the net that is connected to the specified pin (or port). Both `-net` and `-pin` cannot be specified at the same time. The net (or pin) names can be hierarchical names that are unique relative to the top cell.

This command only searches for the defined nets and does not modify any object in database.

Parameters

`{> | >>} filename`	When specified with `>`, writes the report to the specified file name. If the file already exists, the software overwrites it. When specified with `>>`, writes the report to the specified file name. If the file already exists, the software concatenates the report to the end of the file. The *`filename`* parameter must be the last argument in the list. The *`filename`* and `-tcl_list` parameters are mutually exclusive; you cannot specify them together. **Note:** To write a compressed report, add the `.gz` extension to the file name. *Default:* Report is displayed on standard output without being saved.

`-hier`	Reports all of the nets in the design hierarchy. Without this argument, only the nets in the current level of hierarchy are reported.
`-max_fanout` *int*	Reports the net whose number of fanouts is less than or equal to the maximum number specified by *integer*.
`-min_fanout` *int*	Reports the net whose number of fanouts is greater than or equal to the minimum number specified by *integer*.
`-net` *list_of_net_name_or_id*	Reports the set of nets on the list of net names. This argument cannot be specified at the same time as `-pin`. If neither `-net` or `-pin` is specified, all nets are listed by default. **Note:** This argument also reports the capacitance of the net and the input pins.
`-pin` *list_of_pin_name_or_id*	Reports the set of nets to which the pins on the list *list_of_pin_name_or_id* are connected. This argument cannot be specified at the same time as `-net`.
`-tcl_list`	Produces the report in Tcl list format instead of a tabular format. This is useful for extracting information from the report in a Tcl program.

Examples

- Control the `report_nets` table fields, such as setting the minimum and maximum column width using the `set_table_style` command. The table names are `report_nets_source_table`, `report_nets_sink_table`, `report_nets_summary_table`, `report_nets`, and `report_nets_header`.
- For example, the following command sets an indentation of six columns for the table and disables printing of the first column:

```
set_table_style -name report_nets_source_table -indent 6 -max_widths {0,}
report_nets
```

- The format of the `report_nets` report is the same for all the examples, as shown in the first example. The following command reports information about net named `in`, as shown in the following example:

```
report_nets -net n5

Net Name : n5
Number of Sources : 1
Number of Sinks : 1
Number of Bidis : 0
Source of parasitics : Wireload model(.lib None Generated)
Net Capacitance : 0.000
Total Capacitance : 0.011
------------------------------------------------------------------------
Source
------------------------------------------------------------------------
Pin R Driver Load Slew
------------------------------------------------------------------------
Name  Dir  Cell  Cap   Model   Model  Ctotal  Rise   Fall  Phase
-------------------------------------------------------------------------
u5/Q  OUT  DFF   0.000 Unknown CTOTAL 1.000   0.003  0.003 CLK_W_4_gen2(D)(N)
u5/Q  OUT  DFF   0.000 Unknown CTOTAL 1.000   0.003  0.003 CLK_W_4_gen1_inv(D)(P)
-------------------------------------------------------------------------
------------------------------------------------------------------------
Sinks for pin u5/Q
------------------------------------------------------------------------
Pin R Delay Slew
------------------------------------------------------------------------
Name   Dir  Cell  Cap    Parameters Rise   Fall   Rise   Fall   Phase
-------------------------------------------------------------------------------
u7/A  IN   BUF    0.011  -          0.000  0.000  0.003  0.003   CLK_W_4_gen2(D)(N)
u7/A  IN   BUF    0.011  -          0.000  0.000  0.003  0.003   CLK_W_4_gen1_inv(D)
(P)
-------------------------------------------------------------------------------
```

- The following command reports the net to which the pin interesting_pin is connected:

```
report_nets -pin interesting_pin
```

- The following command reports all the nets in the current module:

```
report_nets -net *
```

- The following command reports all nets with a fanout greater than 16:

```
report_nets -net * -min_fanout 16
```

Related Information

- report_path_exceptions
- report_timing

report_path_based_aocv_derate

```
report_path_based_aocv_derate
  [-help]
  [-aocv_derate_only]
  [-delay_corner delayCornerName]
  [-derate_mode {worst_slack worst_derate}]
  [-early]
  [-include_clock_derate]
  [-late]
  [-max_slack val]
  [ -max_paths integer | -max_points integer]
  [-nworst worst_entries]
  [-retime {aocv aocv_path_slew_propagation}]
  [-retime_mode {path exhaustive}]
  [> filename[.gz]]
  [>> filename[.gz]]
```

Parameters

`-aocv_derate_only`	Allows AOCV reporting only.
`-delay_corner` *delayCornerName*	Specifies name of the delay corner.
`-derate_mode` {`worst_slack worst_derate`}	Specifies the derating mode. *Default*: `worst_derate`
`-early`	Specifies derates for early paths.
`-include_clock_derate`	Includes clock derate.
`-late`	Specifies derates for late paths.
`-max_paths` *integer*	Reports derates based on the specified number of paths.
`-max_points` *integer*	Reports derates based on the specified number of endpoints.
`-max_slack` *val*	Reports the upper bound value of the reported slack.
`-nworst` *worst_entries*	Reports the derates based on the specified number of worst paths per end point.
`-retime {aocv aocv_path_slew_propagation}`	
	Reanalyzes the specified set of paths using the specified method. Default: `aocv`
`-retime_mode {path exhaustive}`	Specifies the retime mode to be either `path` or `exhaustive`.
> *filename[.gz]*	Redirects output to the specified file.
>> *filename[.gz]*	Redirects output to the specified file.

Related Commands

- report_timing

report_path_exceptions

```
report_path_exceptions
[-early | -late]
[-ignored]
[-view view_name]
[{> | >>} filename | -tcl_list]
```

Generates a report about path exceptions specified using the `set_false_path`, `set_min_delay`, `set_max_delay` and `set_multicycle_path` commands.

By default, the `report_path_exceptions` command only reports activated path exceptions. To report only the path exceptions that have been ignored by timing analysis, specify the `-ignored` parameter.

If more than one path exception is applied on a timing path, multiple path exceptions that match a given path are prioritized. The adjustment on the path honors the path exception with the highest priority.

When the software is in multi-mode multi-corner timing analysis mode, the report generates path exception information for every active analysis view.

The `report_path_exceptions` command output also includes 2 columns that show the file name and line number. The report output shows the exact file name and line number of the applied/ignored constraints. The compressed constraints will be decompressed first and then shown as individual constraints as written in the SDC file.

Parameters

{> | >>} *filename*	When specified with >, writes the report to the specified file name. If the file already exists, the software overwrites it. When specified with >>, writes the report to the specified file name. If the file already exists, the software concatenates the report to the end of the file. The *filename* parameter must be the last argument in the list. The *filename* and -tcl_list parameters are mutually exclusive; you cannot specify them together. **Note:** To write a compressed report, add the .gz extension to the file name. *Default:* Report is displayed on standard output without being saved.
-early | -late	Generates the path exception report for either early paths or late paths. *Default:* Generates a report based on the current timing analysis mode.

-ignored	Reports only the path exceptions that have been ignored by timing analysis. This parameter can be used for debugging constraints that may be ineffective or incorrectly entered. A path exception may be ineffective if it is covered by, or overridden by, another path exception. For example:
	
	`set_false_path -to OUT`
	`set_multicycle_path -to OUT 2`
	
	In the above scenario, the set_multicycle_path command will be ignored as false paths have higher priority over cycle addition on the same timing paths.

`-tcl_list`	Produces the report in Tcl list format instead of a tabular format. This is useful for integrating timing with custom Tcl functions, and also for customizing report generation. The `-tcl_list` and *`filename`* parameters are mutually exclusive; you cannot specify them together.
`-view` *`view_name`*	Generates a path exception report for the specified analysis view only. You can specify this parameter only when the software is in multi-mode multi-corner timing analysis mode.

Examples

- The following commands shows that a false path has priority over cycle addition on the same path:

```
set_false_path -from I_block/A_reg/Q -to J_block/D_reg/D


set_multicycle_path -from I_block/A_reg/Q -to J_block/D_reg/D 2


report_path_exceptions -early
```

```
+-------------------------------------------------+
|      From       |       To        |     Early     |
|-----------------+-----------------+---------------|
| I_block/A_reg/Q | J_block/D_reg/D | false         |
| I_block/A_reg/Q | J_block/D_reg/D |               |
+-------------------------------------------------+
```

  The term `false` means that the path is treated as a false path due to the path exception, and `add 2 ignored` means that there was a multicycle path constraint but that it is ignored. This is because the false path takes precedence over the multicycle path. If you notice `ignored` path exceptions in your design, check why they are ignored.

- The following command shows how edges are handled:

```
set_false_path -from_rise {in[0]} -through {out[0]}


set_false_path -from_rise {in[0] in[1]} -to_fall {out[0]}}
```

```
set_false_path -from_rise {in[0]}
```

```
report_path_exceptions -early
```

```
+-------------------------------------------------------+
|     From      | Through |    To     |     Early      |
|---------------+---------+-----------+----------------|
| ^ in[0]       | out[0]  |           | false ignored  |
| ^ in[0] in[1] |         | v out[0]  | false ignored  |
| ^ in[0]       |         |           | false          |
+-------------------------------------------------------+
```

```
report_path_exceptions -late
```

```
+-------------------------------------------------------+
|     From      | Through |    To     |     Late       |
|---------------+---------+-----------+----------------|
| ^ in[0]       | out[0]  |           | false ignored  |
| ^ in[0] in[1] |         | v out[0]  | false ignored  |
| ^ in[0]       |         |           | false          |
+-------------------------------------------------------+
```

- When the software is in multi-mode multi-corner timing analysis mode, the software generates path exception information for every active analysis view:

```
+--------------------------------------+
|  From |  To   |     Late       |  View |
|       |       |                |  Name |
|-------+-------+----------------+-------|
| RS/CK |       | add 2          | s2    |
| RS/CK | RT/D  | add 2          | s1    |
| RS/CK |       | add 2 ignored  | h3    |
+--------------------------------------+
```

- Specify the `-view` parameter to generate a report only for a specific active analysis view:

```
report_path_exceptions -view s1
```

```
+-------------------------------+
|  From |  To   |  Late |  View |
|       |       |       |  Name |
|-------+-------+-------+-------|
| RS/CK | RT/D  | add 2 | s1    |
+-------------------------------+
```

Related Information

- report_timing

report_path_groups

```
report_path_groups
[-name group_name]
[{> | >>} filename | -tcl_list]
```

Lists the names of all path groups with the corresponding paths.

Parameters

`{> | >>} filename`	When specified with `>`, writes the report to the specified file name. If the file already exists, the software overwrites it. When specified with `>>`, writes the report to the specified file name. If the file already exists, the software concatenates the report to the end of the file. The *`filename`* parameter must be the last argument in the list. The *`filename`* and `-tcl_list` parameters are mutually exclusive; you cannot specify them together. **Note:** To write a compressed report, add the `.gz` extension to the file name. *Default:* Report is displayed on standard output without being saved.
`-name group_name`	Limits the report to the specified path group. *Default*: All path groups are reported.
`-tcl_list`	Produces the report in Tcl list format instead of a tabular format. This is useful for integrating timing with custom Tcl functions, and also for customizing report generation. The `-tcl_list` and *`filename`* parameters are mutually exclusive; you cannot specify them together.

Examples

The following command shows a sample report:

```
> report_path_groups
---------------------------------
From    To         Group Name
---------------------------------
CLK1    seg1/u3/D  path1
---------------------------------
```

Related Commands

- set_clock_groups
- reset_clock_groups
- reset_path_group
- get_path_groups
- group_path

report_ports

```
report_ports
[-type {[input] | [source_insertion] | [insertion] | [clock_root]
| [uncertainty] | [arrival] | [required] | [external] | [clk_arrival]
| [port_cap] | [fanout_load] | [fanout_load_limit] | [drive_resistance]
| [drive_cell] | [slew_time] | [slew_limit] | [constant] | external_detail
| drive_resistance_detail}]
[-include_pins]
[-pins port_name_list]
[-view view_name]
[> filename | -tcl_list]
```

Reports timing constraints on ports. By default, the command reports all timing constraints on all ports of the current module.

The direction (*DIR*) column of the generated report can contain one of the following abbreviations:

- `IN` indicates that the pin is an input pin.
- `OUT` indicates that the pin is an output pin.
- `INTL` indicates that the pin is an internal pin (a pin inside the instance boundary).

The *Clock Name* column of the report can contain the following symbols:

- `D` indicates it is a data signal in the clock domain.
- `C` indicates it is a clock signal in the clock domain.
- `P` indicates it is a positive phase.
- `N` indicates it is a negative phase.
- Asterisk (`*`) indicates there is a timing exception associated with the signal.

  Note: Currently, it is not possible to expand the asterisk to determine which exception is associated with the reported timing phase.

Parameters

`> filename`	Writes the report to the specified file name. If the file already exists, the software overwrites it. The *filename* parameter must be the last argument in the list. The *filename* and `-tcl_list` parameters are mutually exclusive; you cannot specify them together. **Note:** To write a compressed report, add the `.gz` extension to the file name. *Default:* Report is displayed on standard output without being saved.
`-include_pins`	Reports timing constraints on instance pins, as well as on ports. *Default:* Reports timing constraints on ports only.
`-pins port_name_list`	Specifies the pins or ports to be reported. Pins can be specified by name, or by object ID in a Tcl list.
`-tcl_list`	Produces the report in Tcl list format instead of a tabular format. This is useful for integrating timing with custom Tcl functions, and also for customizing report generation. The `-tcl_list` and *filename* parameters are mutually exclusive; you cannot specify them together.
`-type {assertion_type | assertion_type_list}`	Specifies the constraints to be reported. *Default*: All constraints are reported. The following are valid `assertion_type` values: • `arrival`: Reports arrival time constraints as set by `set_input_delay` command. • `clk_arrival`: Reports clock arrival time constraints as set by the `set_clock_latency` command. • `clock_root`: Reports the clock root as set by the `create_clock` command.

- `constant`: Reports constant constraints as set by the `set_case_analysis` command.
- `drive_cell`: Reports drive cell constraints as set by the `set_driving_cell` command.
- `drive_resistance`: Reports drive resistance constraints as set by the `set_drive` command.
- `drive_resistance_detail`: Reports detail about the input ports with drive resistance constraints.
- `external`: Reports external delay constraints as set by the `set_output_delay` command.
- `external_detail`: Reports detail about the output ports with external delay constraints.
- `fanout_load`: Reports fanout load constraints as set by the `set_fanout_load` command.
- `fanout_load_limit`: Reports fanout load limit constraints as set by the `set_max_fanout` command.
- `input`: Reports the input delay constraints as set by the `set_input_delay` command.
- `insertion`: Reports the network insertion delay as set by the `set_clock_latency` command.
- `port_cap`: Reports port capacitance constraints as set by the `set_load` command.
- `required`: Reports required time constraints as set by the `set_output_delay` command.
- `slew_limit`: Reports slew time limit constraints as set by the `set_max_transition` command.
- `slew_time`: Reports slew time constraints as set by the `set_input_transition` command.
- `source_insertion`: Reports the source insertion

	delay as set by the `set_clock_latency -source` command. • `uncertainty`: Reports the clock uncertainty as set by the `set_clock_uncertainty` command.
`-view` *view_name*	Specifies the view name.

Examples

Here are some examples.

- The following command shows a port report, as shown in the following example:

```
> report_ports

-------------------------------------------------------------------------
Early Late
-------------------------------------------------------------------------
Pin     Dir     Assertion          Clock Name   Rise    Fall   Rise  Fall
Name
-------------------------------------------------------------------------
CLK1    IN      clock_root          CK2(C)(P)
CLK2    IN      clock_root          CLK2(C)(P)
CLK3    IN      clock_root          CLK3(C)(P)
IN1     IN      drive_resistance    *(D)(P)      2.000  2.000  2.000  2.000
IN2     IN      drive_resistance    *(D)(P)      2.000  2.000  2.000  2.000
IN3     IN      drive_cell          *(D)(P)      INV    INV    INV    INV
OUT1    OUT     external            GCLK1(C)(P)  1.000  1.000  1.000  1.000
OUT2    OUT
OUT3    OUT derived_clock GCLK4(C)(P)
-------------------------------------------------------------------------
Pin   Dir    Assertion               Value
Name
-----------------------------------------------------------
CLK3    IN   slew_limit             ( 0.900 : )
CLK3    IN   port_cap_limit         ( 0.020 : 0.900 )
CLK3    IN   fanout_load_limit      5.000
CLK2    IN   slew_limit             ( 0.900 : )
CLK2    IN   port_cap_limit         ( 0.020 : 0.900 )
CLK2    IN   fanout_load_limit      5.000
CLK1    IN   slew_limit             ( 0.900 : )
CLK1    IN   port_cap_limit         ( 0.020 : 0.900 )
CLK1    IN   fanout_load_limit      5.000
```

```
IN3    IN   slew_limit              ( 0.900 : )
IN3    IN   port_cap                ( 1.500 : : 2.600 )
IN3    IN   port_wire_cap           ( 1.200 : : 3.000 )
IN3    IN   port_cap_limit          ( 0.020 : 0.900 )
IN3    IN   fanout_load_limit       5.000

IN2    IN   slew_limit              ( 0.900 : )
IN2    IN   port_cap                ( 1.500 : : 2.600 )
IN2    IN   port_wire_cap           ( 1.200 : : 3.000 )
IN2    IN   port_cap_limit          ( 0.020 : 0.900 )
IN2    IN   fanout_load_limit       5.000
IN1    IN   slew_limit              ( 0.900 : )
IN1    IN   port_cap                ( 1.500 : : 2.600 )
IN1    IN   port_wire_cap           ( 1.200 : : 3.000 )
IN1    IN   port_cap_limit          ( 0.020 : 0.900 )
IN1    IN   fanout_load_limit       5.000
OUT3   OUT  slew_limit              ( 0.900 : )
OUT3   OUT  port_cap_limit          ( 0.020 : 0.900 )
OUT2   OUT  slew_limit              ( 0.900 : )
OUT2   OUT  port_cap_limit          ( 0.020 : 0.900 )
OUT1   OUT  slew_limit              ( 0.900 : )
OUT1   OUT  port_cap_limit          ( 0.020 : 0.900 )
```

- The following command shows `source_insertion` and `uncertainty` assertion type options as shown in the following example:

```
> report_ports -type clock_root source_insertion input -pin clk in
> report_ports -type {uncertainty source_insertion} -pin CLK1

------------------------------------------------------------
Early Late
------------------------------------------------------------
Pin    Dir  Assertion     Clock Rise    Fall  Rise   Fall
Name                     Name
------------------------------------------------------------
CLK1   IN   source_insertion *(C)(P)  -0.500 -0.500 -0.500 -0.500
------------------------------------------------------------

Pin    Dir  Assertion    Value
Name
-----------------------------------------------------------
CLK1  IN   uncertainty  0.400(T) 0.400(T) 0.400(T) 0.400(T)
-----------------------------------------------------------
```

- The following command shows `clock_root` and `uncertainty` assertion type options as shown

in the following example:

```
> report_ports -type {input uncertainty clock_root}

------------------------------------------------------------------
Early Late
------------------------------------------------------------------
Pin    Dir  Assertion   Clock Name    Rise   Fall   Rise   Fall
Name
------------------------------------------------------------------
CLK1   IN   clock_root  CLK_W_1(C)(P)
in1    IN   input       @(D)(P)       1.000  1.000  1.000  1.000
------------------------------------------------------------------

Pin   Dir  Assertion    Value
Name
------------------------------------------------------------
CLK1  IN   uncertainty  0.400(T) 0.400(T) 0.400(T) 0.400(T)
------------------------------------------------------------
```

Related Information

- report_clocks
- report_net
- report_timing

report_socv_interconnect_variation

```
report_socv_interconnect_variation
[-help]
[-delay_corner string]
[-power_domain string]
```

Reports the RC variation multiplier data provided in the SOCV files.

Parameters

`-help`	Prints out the command usage.
`-delay_corner string`	Specifies the name of the delay corner.
`-power_domain string`	Specifies the name of the power domain.

report_socv_library

```
report_socv_library
[-help]
[-categories string]
[-coverage_verbose]
[-list_not_annotated]
[-library_set string | { -delay_corner string [ -power_domain string] }| { -view string [-
power_domain string]}]
[-libraries string | -lib_cells string]
{-coverage}
```

Reports the coverage of SOCV library(s) loaded in the design.

Parameters

`-help`	Prints out the command usage.
`-categories` *string*	Allows you to control what types of table data to check delay arcs, transition arcs, and check arcs. Default is all.
`-coverage`	Control for reporting coverage library data.
`-coverage_verbose`	Generates a detailed report table per cell.
`-delay_corner` *string*	Checks all the libraries defined in the specified library set.
`-lib_cells` *string*	Specifies the library cells. The allowed format is \u201clibrary/cell\u201d name or pattern or collection. This will typically be done to drill-down debugging of problems.
`-libraries` *string*	Allows you to select the library. If -library_set, -delay_corner, or \u2013view is not specified then all libraries matching the library name across the MMMC configuration will get checked.
`-library_set` *string*	Checks all the libraries defined in the specified library set.
`-list_not_annotated`	Allows you to control the types of table data to check delay arcs, transition arcs, and check arcs. Default: all
`-power_domain` *string*	Checks the specified power domain.
`-view` *string*	Checks all the delay corners associated with the specified view.

report_statistical_timing_derate_factors

```
report_statistical_timing_derate_factors
[-help]
[-derate_cell]
[-derate_cell_check]
[-derate_net]
[-max_slack float]
[-path_based_ssta]
[-path_group groupname_list]
[-ssta_view ssta_view_name]
[-sta_view sta_view_name]
[-type {arrival_time_based | slack_time_based}]
[-early | -late ]
[-clock_factor | -data_factor]
[-max_paths integer | -max_points integer]
```

Generates design specific OCV factor. The scaling factor (hold or setup) is computed either by using the arrival time or slack time approach.

Parameters

`-clock_factor | -data_factor`	Specifies if the user-defined factor value is for clock or data. If you specify the `-clock_factor` parameter, then the `-data_factor` parameter is calculated by the software, and vice versa. These parameters are applicable to type `slack_time_based`. **Note:** The `-clock_factor` parameter and the `-data_factor` parameter are mutually exclusive. You cannot specify them together.
`-derate_cell`	Applies user-specified scaling factors on cell delays.
`-derate_cell_check`	Applies user-specified scaling factors on timing checks.
`-derate_net`	Applies user-specified scaling factors on net delays.

`-late	-early`	Specifies early or late mode analysis. Default is `late`.
`-max_paths`	Reports the specified number of worst paths in the design, regardless of the endpoint.
`-max_points` *integer*	Reports the worst path that is found to each endpoint up to the number specified by the `-max_points` option.
`-max_slack` *float*	Reports only those paths with slack equal to or less than the specified float value.
`-path_based_ssta`	Enables generation of the statistical OCV factor using path based SSTA analysis.
`-path_group` *groupname_list*	Considers only the path contained in the groups in the specified *groupname_list* for computing the statistical timing derate factor.
`-ssta_view` *ssta_view_name*	Specifies view for SSTA run.
`-sta_view` *sta_view_name*	Specifies view for STA run.
`-type arrival_time_based	slack_time_based`	Specifies if the OCV factor needs to be generated using the arrival time or slack time approach. In the arrival time approach, the SSTA arrival time values are used to compute the statistical OCV factor for clock and data. In the slack time approach, the STA and SSTA slacks are used to generate the statistical OCV factor for either clock or data depending on whether you specify the factor for data or clock. *Default:* `arrival_time_based` You can use the `-type arrival_time_based` parameter with the following parameters only: • `-late` • `-early` • `-path_based_ssta`

	You can use the `-type slack_time_based` parameter with the following parameters only: • `-ssta_view` • `-sta_view` • `-late` • `-early` • `-clock_factor` • `-data_factor` • `-derate_cell` • `-derate_net` • `-derate_cell_check` • `-max_paths` • `-max_slack`

Examples

- The following example computes the factors for setup analysis using the arrival time based approach:
  ```
  report_statistical_timing_derate_factors -type arrival_time_based -late
  ```
- The following example computes the data factor for setup analysis using the slack time based approach applying a user defined clock factor of 0.95:
  ```
  report_statistical_timing_derate_factors -type slack_time_based -ssta_view viewname -sta_view viewname -clock_factor 0.95
  ```
- The following example computes the clock factor for hold analysis using the slack time based approach applying a user defined data factor of 1.05:
  ```
  report_statistical_timing_derate_factors -type slack_time_based -ssta_view viewname -sta_view viewname -data_factor 1.05 -early
  ```

report_timing

```
report_timing
[-help]
[-analysis_summary_csv_extended_file <string>]
[-analysis_summary_csv_file <string>]
[-analysis_summary_file <string>]
[-check_clocks]
[-check_type {setup hold clock_gating_setup clock_gating_hold clock_gating_pulse_width
data_setup data_hold recovery removal pulse_width clock_period clock_separation skew}]
[-debug {unconstrained | time_borrow | cppr_point}]
[-derate_summary]
[-gui]
[-fields <column_list>]
[-group <groupname_list>]
[-hpin]
[-max_delay <float>]
[-max_slack <float>]
[-min_slack <float>]
[-nets]
[-output_format {text | csv | gtd | binary}]
[-path_exceptions {applied | ignored | all}]
[-path_type {endpoint | summary | full | full_clock}]
[-retime {aocv  | path_slew_propagation | aocv_path_slew_propagation}]
[-retime_format {manual | retime_compare | retime_replace}]
[-retime_mode {path | exhaustive}]
[-skip_io_paths]
[-split_delay]
[-unique_pins]
[-view <viewName>]
[> <filename[.gz]>]
[>> <filename[.gz]>]
[-late | -early]
[-rise | -fall]
[-begin_end_pair | [[ -max_paths <integer>] [-nworst <integer>]]]
[-unconstrained | -point_to_point]
[-collection | -tcl_list ]
[-not_through <object_list> | -not_rise_through <object_list> | -not_fall_through
<object_list>]
[ [-from <pin_list> | -from_rise <pin_list> | -from_fall <pin_list>] [-through <pin_list>
| -through_rise <pin_list> | -through_fall <pin_list>] [-to <pin_list> | -to_rise
<pin_list> | -to_fall <pin_list>] ]
```

Generates a timing report that provides information about the various paths in the design.

The reports typically contain data on the delay through the entire path. The start node and the end node of each path is identified.

The timing report contains the following information:

- The slack times of the arriving signal at the end node
- The start node
- The associated transitions
- The signal required times and the actual signal arrival times
- Summary of propagated vs ideal status of launching and capturing clocks
- Any phase shifts applied when evaluating timing checks
- Any CPPR values applied to timing check evaluation

The `report_timing` command output accommodates both the instance pin names and net names in the same column. Since both instance pin names and net names can be long, to save screen space alternating instance pin and net names (depending on whether the row displays gate delay or net delay) are reported in a single column.

When the `timing_report_timing_header_detail_info` timing attribute is set to `extended`, the software generates timing reports with additional header information, that includes the following details:

- The analysis mode - best case worst case or on chip variation
- The active views (displayed only in MMMC mode)
- The clock reconvergence pessimism (CPPR)
- The latch slack mode
- The specified clock propagation value
- The delay calculation
- The specified delay calculation engine

When paths are grouped by clock domain, the `report_timing` command will have the following four types of groups:

- Clock gating group: This is an internal group. This group reports the clock gating check type paths.
- Asynchronous group: This is also an internal group and reports the asynchronous check type paths.

- Groups for all the clocks in the design: These are also internal groups, classified on the basis of each reference clock. Paths belonging to each reference clock are reported separately under each clock name's group. These groups do not honor the clock gating/asynchronous check types. Suppose there are three clocks C1, C2, and C3 in the design, then three groups will be reported for each of these clocks. The endpoint sharing the same reference clock will be part of the same group. In case the endpoint belongs to multiple groups then it will be reported separately in each group.
- Default group: This is an internal group. Combinational max delay or min delay paths.
- User-Defined Groups: These are user-defined path groups. These are created by users with single or multiple constraints. These groups honor all the check types. These groups are now reported by default.

Note: The attribute does not honor user-defined path groups so the paths will still be grouped by different clocks.

The `report_timing` command supports multi-threaded reporting. You can make the following settings to enable multi-threaded reporting:

`set_multi_cpu_usage -localCpu` *`number_of_CPU`*

Note: When `timing_allow_input_delay_on_clock_source` attribute is enabled and the `set_input_delay` command is applied to a clock input, the specified delay value will be considered as the source clock latency for clock paths and as regular data path arrival time for clock source paths.

This page provides the following `report_timing` details:

- Parameters
- Use of Informational Flags
- Reporting Timing Checks Between Various Types of Clocks
- Retime Analysis Report Header Description
- Examples
- Sample Reports
- Path Summary Report
- Advanced Summary Section
- Path Description Section
- End Point Reports
- Clock Tracing Reports

- Related Information

Parameters

`{> | >>} filename`	When specified with >, writes the report to the specified file name. If the file already exists, the software overwrites it. When specified with >>, writes the report to the specified file name. If the file already exists, the software concatenates the report to the end of the file. The *filename* parameter must be the last argument in the list. The *filename* and -tcl_list parameters are mutually exclusive; you cannot specify them together. **Note:** To write a compressed report, add the .gz extension to the file name. *Default:* Report is displayed on standard output without being saved. **Note:** The last argument specified in the `report_timing` command is not considered a filename.
`-analysis_summary_csv_extended_file <string>`	Generates the CSV file to be used for merging.
`-analysis_summary_csv_file <string>`	Generates the analysis summary CSV file.
`-analysis_summary_file <string>`	Generates the analysis summary file.

`-begin_end_pair`	Reports all violating paths between unique source and target register pairs. To report other paths, specify this parameter with the `-max_slack` parameter. Paths are not sorted by slack; instead, paths with the same source pins are reported adjacent to each other. When specified with the `-from`, `-through`, or `-to` parameters, the software only reports the unique register pair paths that match the path description. **Note:** You cannot specify this parameter with the `-max_paths`, `-nworst`, `-unique_pins`, or `-min_slack` parameters.
`-check_clocks`	Generates reports based on timing paths on the clock network instead of the standard timing to data endpoints. This report includes clock paths that end at the reference end of a check or a clock gating end point. *Default*: Data paths and all the other clock paths (for example, a clock path ending at a D pin of a register) are reported.
`-check_type` *check_type*	Reports only the paths that end at the specified timing check. The following check types can be reported when the software is in setup analysis mode: setup, skew, pulse_width, clock_period, clock_gating_setup, clock_gating_pulse_width, data_setup, recovery, clock_separation, and no_change_setup. The following check types can be reported when the software is in hold analysis mode: hold, clock_gating_hold, data_hold, removal, and no_change_hold. If the software is in simultaneous setup and hold analysis mode, all of the check types listed above can be reported.

	By default, the following checks are reported: • setup • hold • clock_gating_setup • clock_gating_hold • data_setup • data_hold • recovery • removal • no_change_setup • no_change_hold You can use the `-check_type` parameter to specify the following clock style checks: • pulse_width • clock_gating_pulse_width • clock_period • Skew • clock_separation Do not use this parameter with the `-unconstrained`, `-early`,or `-late` parameters. **Note:** The `report_timing` command does not perform skew checks by default.
`-collection`	Returns a collection of timing paths. This is useful for performing Tcl queries on selected timing reports. To generate a collection of paths analyzed by Path Based Analysis (PBA), you can use the `-retime` parameter. **Note**: This parameter is not supported in DMMMC mode.

`-debug {unconstrained | time_borrow | cppr_point}`	You can specify the following options for debugging: • `unconstrained`: Determines the reasons for a path to be unconstrained. The report output will list all the applicable arcs/constraints that could not constrain the paths, along with the reasons. • `time_borrow`: Reports time borrowing calculations. Time borrowing is the amount of time borrowed by a previous logic. • `cppr_point`: Reports the CPPR point information for the reported paths.
`-derate_summary`	Generates an AOCV derating summary table for the launch and capture paths, in addition to the timing report.
`-gui`	Displays the timing report in a GUI window.
`-early | -late`	Generates the timing report for early paths (hold checks) or late paths (setup checks). *Default*: `-late` **Note:** If you specify a check, setup or hold, using the set_analysis_mode command, the `report_timing` command reports that check by default.
`-fields` *column_list*	Formats the report according to the *column_list*. The *column_list* specifies which columns to display in the timing report and the order in which they appear. You can use the `-fields` parameter to customize the reports to your needs by requesting the exact fields in which you have an interest. The valid field columns are: adjustment, abs_delay_err, abs_transition_err, annotation, aocv_adj_stages, aocv_derate, aocv_weight, arc, arrival, arrival_mean,

arrival_sigma, cell, delay, delay_mean, delay_sigma, direction, edge, fanin, fanout, flags, hpin, incr_delay, instance, instance_location, latch_window, load, module, net, pct_delay_err, pct_slew_err, phase, phys_info, pin, pin_load, pin_location, power_domain, required, retime_delay, retime_delay_mean, retime_delay_sigma, retime_slew, retime_transition_mean, retime_transition_sigma, stage_count, transition, transition_mean, transition_sigma, socv_derate, spice_arrival, spice_delay, spice_slew, stolen, timing_point, total_derate, user_derate, voltage, when_cond, wire_load, wlmodel

For more details on the 'flags' column, refer to the section on Use of Informational Flags.

For example:

```
-fields {hpin cell delay required arrival required edge}
```

Default: `{timing_point flags arc edge cell power_domain fanout transition delay arrival}`

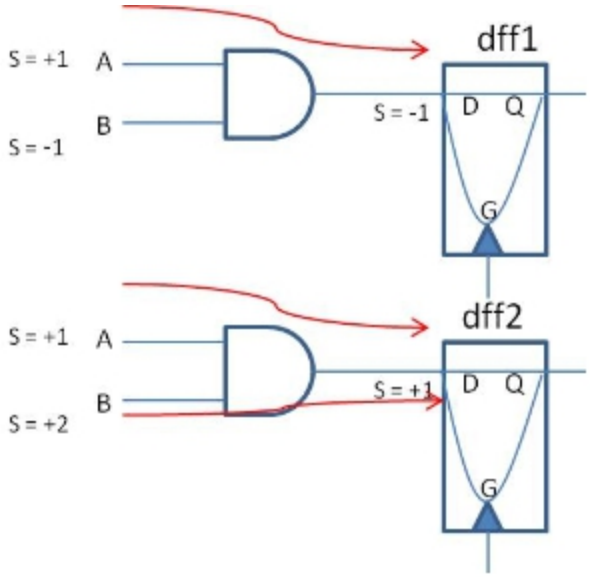
See Table for a list of valid options.

The default options can be changed when the following parameters are specified:

- The default net format (with `-net` option) for the full path is:
  ```
  {hpin edge net cell delay arrival required}
  ```
- If the `-unconstrained` option is specified, then the required column is not displayed.
- The adjustment column is automatically added to the timing report. To disable this feature you can set the `report_timing_default_adjustment_column` attribute to `false`.
- You can also use a combination of the `-fields`

	and `-tcl_list` parameters to integrate the timing reports into your Tcl scripts. You can use the `-from` parameter to limit the number of paths reported, and to find specific paths in the design. • If you specify the `report_timing -split_delay` parameter, then the timing report format will change to: `{timing_point flags edge cell power_domain fanout transition delay arrival}`
`{-from | -from_rise | -from_fall}` *`pin_list`*	Reports paths starting from the object(s) specified in the *`pin_list`*. The supported objects are pins, ports, and clocks. Using `-from_rise` (or `-from_fall`) parameter specifies that the rising (or falling) edge of the signals on the objects in the *`pin_list`* are the start of the paths. This option also reports on source clock waveform(s). You can use this option with the `-through` and `-to` parameters for specifying particular paths in the design.
`-group` *`groupname_list`*	Reports only paths contained in the groups specified in the *groupname_list*.
`-hpin`	Prints all the hierarchical crossings of an arc under the hpin column in the `report_timing` command table.
`-max_delay` *`float`*	Specifies the path delay limit for unconstrained paths.

`-max_paths` *integer*	Reports the specified number of worst paths in the design, regardless of the endpoint. This is useful, but can be time consuming if a large number of paths are requested. The paths are always sorted based on slack. *Default*: worst path When `timing_report_group_based_mode` attribute is set to true, the `-max_paths` parameter reports the specified number of paths per clock group.
`-max_slack` *float*	Reports only those paths with slack less than the value of the *float*. The `-max_slack` option limits the report to paths that fall into the specified range. A positive slack value indicates that timing was met. A negative value for slack indicates a timing violation. The `-max_slack` option cannot be used with the `-unconstrained` option.
`-min_slack` *float*	Reports only those paths whose slack is greater than the value of float. The `-min_slack` option cannot be used with the `-unconstrained` option. You can generate a timing report showing paths with slack greater than the specified slack by using the `-max_paths` option. For example, to report all paths with slack greater than 2ns, you can issue the following command: `min_slack 2.0 -max_paths 1000` In this case the maximum number of paths reported will be 1000. When the `-min_slack` parameter is specified, the total numbers of paths processed will be as determined by the `max_path`s/`nworst` settings and only those paths satisfying the `-min_slack` criteria will be reported. The software will not process paths beyond the `max_paths`/`nworst` limit to find

	other paths satisfying the `-min_slack` limit. In the below diagram, if `-min_slack` is 0 and `-max_paths 4 -nworst 2` is specified, then only 3 paths will be reported; path with slack value of -1 will be ignored. Here, even if there is another path with slack that is greater than or equal to 2, then it will not be reported because worst max_paths have been processed. 
`-nets`	Adds a row to the "Timing Point" field column to display the net names. **Note**: The `-split_delay` and `-nets` parameters can be used together.

{-not_through | -not_rise_through | -not_fall_through} *object_list*	-not_through: Reports paths that do not traverse through the specified nets, ports, or pins of a cell. By default the parameter will be ignored when an object has already been specified. In this case the parameter shows the following behavior: • If a collection of all the input ports has been specified (using all_inputs command), then the report will show paths to the inout part, and paths starting from the internal part of inout will be excluded. • If a collection of all output ports has been specified (using the all_outputs command), then the report will show paths starting from the internal part of inout, and paths ending at inout part will be excluded. • If any other collection (not specified using all_inputs and all_outputs commands) is specified, then paths to both the parts will not be reported.
	-not_rise_through: Excludes paths with rise transition.
	-not_fall_through: Excludes paths with fall transition.
-nworst *integer*	Specifies the number of paths to be enumerated for each endpoint. Use the -nworst option to report all the checks at an end point or use the -check_type option to report a specific check. *Default*: Only the worst path to each endpoint is reported. When timing_report_group_based_mode attribute is set to true, the -nworst parameter limits the number of paths per endpoint per clock group. In this mode, the default value of -nworst is 1.

`-output_format {text | csv | gtd | binary}`	Generates a detailed timing report in the specified format - text, CSV, GTD, or binary.

`-path_exceptions {applied | ignored | all}`	Includes information of path exceptions applied and considered for the reported path. If multiple from, through, or to pins are specfied, the exception report for that path will show only those pins that are part of the current path. **Note:** You cannot use this parameter with the following report_timing parameters: • `-point_to_point` • `-collection` • `-path_type summary` • `-tcl_list` In addition, you cannot use this parameter with block-based SSTA. **Note**: If a path is already unconstrained, then all further path exceptions applied on that particular path will be reported as ignored exceptions. The `-path_exception` parameter also reports -start/-end options with multi-cycle paths. Without start or end specification, by default, the setup check is relative to the end clock and the hold check is relative to the start clock. The `report_timing -path_exception all` command will report all the non-default -start/-end specifications. You can specify the following options: • applied: Reports the honored exception on the reported path. • ignored: Reports the ignored exceptions on the reported path. • all: Reports both applied and ignored set of exceptions on the reported. **Note:** Analyzing all the exceptions for a path will cause increase in runtime.

`-path_type {endpoint | summary | full | full_clock}`	The path_type option lets you choose the format of the report by path type. The default format, if the `-path_type` option is not specified is `full`. Choose one of the following: • `endpoint`: Generates an end point report for each path containing the ID, slack, endpoint, and group view. This option generates a very fast report. • `summary`: Reports the detailed summary format same as the header of `path_type full`. • `full`: Generates a report that displays the full path with accompanying required time and slack calculation. This is the default path type. • `full_clock`: If the path reported ends at a timing check, this option also reports the full clock path (Other End Path) in addition to the full data path (Timing Path).

`-point_to_point`	Traces the worst delay path between a pair of from-to pins. Paths reported with this parameter only include the cumulative delay of the path. You must specify either the -from or -to parameter, to define the begin and end points of the path. If you do not specify a -from pin, the standard definition of the begin point of a path applies. If you specify a `-from` pin, it is treated as the begin point of the path. If you specify a -from pin that is not a valid begin point, the arrival time of the first pin starts from 0. If you do not specify a -to pin, the standard definition of the end point applies. If you specify a -to pin, it is treated as the end point, and only those paths ending in -to pins are reported. If you specify more than one -to pin, the worst path to each pin-to-pin is reported. When from-to pins are specified, the software will trace across the trigger arcs and report a path, if there is a clock reaching the CK pin of the trigger arc. However, if there is a begin or end point present between the -from and -to pins, the software will not trace beyond it. If there is no clock reaching the CK pin of the trigger arc, the software cannot trace across the trigger arc, and no path is reported. Additionally, clock path delay is included only when the clock is in propagated mode. **Note:** The `-point_to_point` and `-through` parameters are mutually exclusive; you cannot specify them together. **Note**: The `-point_to_point` parameter supports a single `-from` and `-to` pin; a list of pins is not supported. See Sample Reports.

`-retime_format {manual | retime_compare | retime_replace}`	Allows you to set the format of timing reports for retime related fields. You can specify one of the following options: • `manual`: Does not change the `report_timing` format. • `retime_compare`: If GBA columns are present in the report format, then the corresponding retime options are added. If PBA columns are already present, then there is no change in the format. • `retime_replace`: If GBA columns are present in the report format, then the GBA columns are replaced by the corresponding PBA columns. If PBA columns are already in the report, then there is no change in the format. **Note**: The timing report format is also controlled using the `timing_report_retime_formatting_mode` attribute. The `report_timing` command setting has precedence over this attribute.

`-retime_mode {path | exhaustive}`	Reanalyzes the specified set of paths using the specified method. You can use one of the following options: • `path`: Reanalyzes the specified set of paths. By default, the software uses `-retime_mode path` setting, when the `-retime path_slew_propagation` option is specified. • `exhaustive`: Enables exhaustive path-based analysis mode. In this mode, the software will exhaustively examine nworst paths of the selected endpoints and then select the true worst path to be reported - based on retimed slacks of all the examined paths. When you use this option, reporting can be run time intensive as the software needs to do an exhaustive search for true worst paths. Therefore, it is recommended to use this option in the later phases of the sign-off stage when there are very few violations left for analysis. *Default*: `path`
`-retime {aocv | path_slew_propagation | aocv_path_slew_propagation}`	Reanalyzes the specified set of paths using path based analysis (1 path per endpoint). **Note:** This feature is not meant to be run on all the paths of a design.

	aocv: Computes AOCV stage count based on specific paths and applies the AOCV derate values to GBA (graph-based analysis) delays. The `-retime aocv` option can be used with the following parameters only: • `-rise/-fall` • `-early/-late` • `-check_type` • `-max_paths` • `-begin_end_pair` • `-from/-from_rise/-from_fall` • `-through/-through_rise/ -through_fall` • `-to/-to_rise/-to_fall`
	• `-path_exceptions` • `-path_type` • `-max_slack` • `-min_slack` • `-view` • `-derate_summary` • `-nworst`
	`path_slew_propagation`: Re-evaluates the given set of graph-based analysis (GBA) paths by recalculating the delay values based upon actual propagated slew across the path. In this case the AOCV derates are the ones that are reported during graph-based analysis. The default timing analysis uses worst-case slew propagation to calculate the delays at each stage. As a result, timing analysis results can be pessimistic.

Note: This feature should be run only on paths that are not meeting the slack margin criteria, to help identify actual failing paths from those that are simply failing due to the conservative delay calculation.

Signal integrity incremental delays can be recomputed for the entire path. You can use the following command to perform path based signal integrity analysis:

```
report_timing -retime path_slew_propagation
```

In the report_timing command output, the Retime Incr Delay column reports recomputed signal integrity delays.

The `-retime` option also supports retiming a collection of paths. This option does not retiming an already retimed path collection. This feature works in graph-based analysis (GBA) mode for path collections. You can use the following use model to retime collections:

```
report_timing $collection -retime
path_slew_propagation
```

A path collection stores context dependent data, that is, a delay property in GBA will reflect delay data of the timing path report generated in GBA mode. The same property in PBA will reflect the re-time delay data of the timing path report in PBA mode.

The delay column in timing reports is also context dependent - under the `timing_report_retime_formatting_mode retime_replace` global variable.

Note: For memory optimization purposes, both the delays will not be stored in the same path collection. You can use this setting for comparison purposes.

	`aocv_path_slew_propagation`: Re-evaluates the given set of graph-based analysis (GBA) paths by using path-based AOCV derates and delays using actual path- based propagated slew. This option is a combination of `-aocv` and `-path_slew_propagation`. For retime report header details, see Retime Analysis Report Header Description.
`-skip_io_paths`	Specifies whether the input/output paths are to be ignored.
`-rise | -fall`	Reports the path with the specified edge on the endpoint. If an endpoint is specified using -to_rise (or -to_fall) option, the -rise (or -fall) option is ignored and paths with edge specified by `-to_rise` (or `-to_fall`) are reported. The `-rise` option applies to end point only.
`-split_delay`	Adds a row for the net arc. **Note**: The `-split_delay` and `-nets` parameters can be used together.
`-tcl_list`	Produces the report in Tcl list format instead of a tabular format. This is useful for integrating timing with custom Tcl functions, and also for customizing report generation. The -tcl_list and *filename* parameters are mutually exclusive; you cannot specify them together.

`-through	-through_rise	-through_fall` *pin_list*	Reports paths that pass through pin(s) or net(s) specified by the *pin_list*. Any number of -through pins or nets can be specified. Using -through_rise (or -through_fall) specifies that the paths go through the rising (or falling) edge of the signals on the pins/nets in *pin_list*. The *pin_list* is a logical OR function. The resulting path may pass through any of the pins or nets in the -through *pin_list*. To force the report to pass through multiple pins or nets, separate -through statements are needed. You can use these parameters with the `-net` option.
`-to	-to_rise	-to_fall` *pin_list*	Reports paths leading to the pin(s) specified by the *pin_list*. Pins in the *pin_list* can be either pins on the design boundary (ports) or pins on an instance. Only one list of -to pins can be specified per report. Using -to_rise (or -to_fall) specifies that the rising (or falling) edge of the signals on the pins in *pin_list* are at the end of the paths. This option also reports on target clock waveform(s). The `-to_rise` option applies to end point as well as intermediate pins.
`-unconstrained`	Reports only the unconstrained paths (paths with no slack). Each signal arriving at the path end node which does not have a matching required time, results in an unconstrained path. The `report_timing` command without the `-unconstrained` option reports only constrained paths. If no constrained path is found, there may be unconstrained paths or the path may not exist. **Note**: The `-min_slack` and `-max_slack` cannot be specified with the `-unconstrained` parameter. See Sample Reports for the conditions under which a path is reported as constrained or unconstrained.

`-unique_pins`	Reports paths through unique set of pins. When this parameter is specified, in group-based mode for a given begin-end point pair, the software will consider all different launch clock paths for reporting unique paths. If for a given begin-end pair and a given capture clock, there is a possibility of different launch clock paths, then each such path will be reported. **Note**: Using this option may significantly affect run time and memory usage if the number of paths to be reported is high.
`-view {`*`viewName`*`}`	Generates report for the specified analysis view. You must be in multi-mode multi-corner analysis mode in order to use this parameter. To create a view, you use the `create_analysis_view` command. To set an active view for analysis, you use the `set_analysis_view` command. *Default:* Reports the worst path to each endpoint across all analysis views. The report_timing command to performs block based analysis in MMMC mode. By default, the command reports the worst end-point(s) across all views. You can use the -view parameter to report for individual views.

Use of Informational Flags

Informational tags are used providing additional details. The following informational tags are supported:

Flags	Description
(i)	Net is ideal
(P)	Instance is preserved / don’t_touch

(p)	Instance is preserved / don’t_touch but may be resized

The output format uses the flag keys on the timing report and adds a description at the end of the report for reference. A sample output is as shown below:

```
#-------------------------------------------------------------------------
# Timing Point  Delay (ns) Latch  Window   Arrival   Arc   Flags   Adjustment
# (ns)
#-------------------------------------------------------------------------
f0/CP           -          -    1.000000            CP       -      -
f0/Q            0.254100   -    1.254100            CP->Q    -      -
b32/Z           0.598446   -    1.852546            I->Z     -      -
...
b4/Z            0.086940   -    2.502181            I->Z     -      -
f1/D            0.000000   -    2.502181            D        <<<    -
#-------------------------------------------------------------------------

(P) : Instance is preserved
(p) : Instance is preserved but may be resized
(i) : Net is ideal.
(b) : Timing paths are broken.
```

Report Timing- Column List Options

Option	Description

adjustment	Reports generated clock adjustment values. When reporting the detailed path for a generated clock, there are circumstances where the arrival time needs to be adjusted when a `create_generated_clock` assertion is encountered along the path. The adjustment argument can be used to provide a clearer indication of why a "jump" in the arrival time is seen in the path report. There are two primary reasons for generated clock-related adjustment to occur. • If the `timing_enable_generated_clock_edge_based_source_latency` attribute is set to `false`, it is possible that the edge-to-edge requirement specified by the generated clock cannot be satisfied by an existing logic path. This can happen when the source clock to generated clock polarity of the constraint is non-inverting, but the clock path itself constrains an inverting path. In this case, it would be normal to see a half-cycle of adjustment as the software compensates for the path versus constraint mismatch. If the above mentioned attribute is set to true, the software handles the path versus constraint mismatch by zeroing the source latency of the generated clock, and forcing alignment of the source and generated clock. As a result, no further adjustment is required, or reported. • In cases where `create_generated_clock -edge_shift` is specified, the adjustment column will show the amount of shift specified.
abs_delay_err	Reports the absolute delay difference of Innovus from Spice.

abs_transition_err	Reports the absolute slew difference of Innovus from Spice.
annotation	Reports RC parasitic annotations for net arcs, and delay annotations for either gate or net arcs. The following annotations are reported: • SDF: SDF back-annotation. The annotation status reflects the gate arc, not the interconnect arc. • SPEF: This entry appears if parasitic extraction has been done or annotated using SPEF. • LumpedRC: Lumped RC annotation • DlyAssert: Assertion set with set_annotated_delay command • WLM: Wire load model back-annotation • <none>: All others
aocv_adj_stages	Shows up "Aocv Adjust Stages" column. This column reports the AOCV adjustment stages that indicate the total effective stage count.
aocv_derate	Reports the graph-based AOCV derating factor used for cells/nets in AOCV mode. In retime mode, this column will report graph-based derating for cells/nets before the AOCV branch point and path-based derating.
aocv_weight	Reports AOCV weight values for the path elements.
arc	Reports the arc as described by the from pin, from pin edge, to pin, and to pin edge. For example, the arc from the rising edge of pin A to the falling edge of pin Z is reported as `A -> Z`.
arrival	Reports the arrival time on the pin.
arrival_mean	Reports mean of the arrival time on a pin.

arrival_sigma	Reports standard deviation of the arrival time on a pin.
cell	Reports the cell name of the given pin's instance.
delay	Reports the arc delay. If the stolen slack at the arc output pin (output of transparent latches) is not zero, the stolen column is also displayed along with this column.
delay_mean	Reports the mean of each arc's delay.
delay_sigma	Reports the standard deviation of each arc's delay. **Note**: In the absence of correlated and uncorrelated components of variation, the individual delay sigma values will not be summed up.
direction	Reports the pin direction (IN, OUT).
edge	Reports the edge of a pin (^=R, v=F).
fanin	Reports the number of source nodes of a net connected to a timing pin.
fanout	Reports the number of sinks of a net connected to a timing pin.
flags	Indicates the size_only, dont_use, dont_touch, and ideal attributes. When the 'flags' option is specified, a capacitance group entry (if it exists) will be displayed under the "Flags" column. This value indicates the input capacitance group being used for delay calculations – C1C2 (2-piece model) or C1CN (n-piece model).
hpin	Reports the hierarchical name for a given pin. **Note**: The hpin, pin, and timing_point will print the same values in the report output.

incr_delay	Reports the incremental delay of a pertaining arc. The software reads the incremental delay from an incremental SDF file using the `read_sdf` command.
instance	Reports the hierarchical name of a given pin's instance.
instance_location	Reports the location (x,y) of an instance. If a design is not placed, this column will be blank. This option displays the `instance` column automatically.
latch_window	Reports the available window for latch to sample the data.
load	Reports the total capacitance load on a given pin.
module	Reports the module name of the corresponding block hierarchy. This hierarchical module name will be reported only when the `-hpin` parameter of the `report_timing` command is specified.
net	Reports the hierarchical name of a net connected to a given pin.
pct_delay_err	Reports the percent delay difference of Innovus from Spice.
pct_slew_err	Reports the percent slew difference of Innovus from Spice.
phase	Reports the phase name of a pin.
phys_info	Shows minimum/maximum assigned layer, and non-default rule name information.
pin	Reports the reference name for the given pin.
pin_load	Reports the pin load data. The pin load refers to the sum of pin capacitance of all the pins connected to the current net.
pin_location	Reports the location (x,y) of a pin. If a design is not placed, this column will be blank.

power_domain	Reports power domain details of the specified path instance.
required	Reports the required time on a pin.
retime_delay	Reports the recalculated arc delay based on the path being reanalyzed with the specified retime method.
retime_delay_mean	Reports the mean of the recalculated arc delay on the path being reanalyzed with the specified retime method.
retime_delay_sigma	Reports the standard deviation of each parameter of an arc's retimed delay.
retime_slew	Reports the recalculated slew based on the path being reanalyzed with the specified retime method.
retime_slew_mean	Reports the mean of each parameter of an arc sink pin's retimed slew.
retime_slew_sigma	Reports the standard deviation of each parameter of an arc sink pin's retimed slew.
stage_count	Displays the aggregate stage count of a specific instance in graph-based analysis (GBA) mode, and the actual stage count in path-based analysis (PBA) mode.
transition	Reports the propagated slew at the given pin.
transition_mean	Reports mean of each parameter of an arc sink pin's slew.
transition_sigma	Reports the standard deviation of each parameter of an arc sink pin's slew.
socv_derate	Reports the SOCV or the spatial derating factor applied on the cells or nets in SOCV analysis mode.
spice_arrival	Reports the Spice arrival time at the timing point.
spice_delay	Reports the Spice delay of a timing arc.

spice_slew	Reports the Spice slew of a timing point.
stolen	Reports the slack stolen (or the time given to the previous stage) at the given pin. If the slack is not zero and the `delay` column is specified, then this column is displayed by default (but only visible on the output of transparent latches).
timing_point	Reports the hierarchical names for the given points.
total_derate	Reports the total derating factor applied on the corresponding cell or nets that reflect the combined impact of all the derates (OCV/AOCV/spatial) applied on each stage of the timing path.
user_derate	Reports the derating scale factors set with the `set_timing_derate` command. **Note:** The timing system cannot provide an accurate derating factor for the lumped gate and wire delay when the `set_timing_derate` factors for `-cell_delay` and `-net_delay` are not equivalent. In this case, an asterisk (*) is reported in the SSI Derate column.
voltage	Returns the voltage that is used during delay calculations.
when_cond	Reports the "when" condition of the arc specified in the library.
wire_load	Reports the wire load.
wlmodel	Reports the kind of parasitic used by a pin or net, such as SPEF. If the wire load model is derived from a Liberty file, the information is reported in the following format: `Wireload model (.lib` *`model_name library_name`*`)`

Reporting Timing Checks Between Various Types of Clocks

The following table summarizes the timing checks and the min/max delay checks that are performed based on the types of launching and capturing clocks:

- NZ (Non Zero Period Clock - Regular Clock)
- Z (Zero Period Clock - Regular clock whose period is equal to zero)
- @ (Asynchronous Clock - Clock which triggers the default input delay and the input delay without a clock)
- CPD (Combinational Path Delay)

Launching Clock	Capturing Clock	Timing Checks	Path Delay Checks
@	@	NO	CPD
Z	@	NO	CPD
NZ	@	NO	CPD
@	Z	YES	SPD
@	NZ	YES	SPD
NZ	Z	NO	SPD
Z	NZ	YES	SPD
Z	Z	YES	SPD
NZ	NZ	YES	SPD

Retime Analysis Report Header Description

AOCV(default)	Indicates that the selected retime analysis includes recalculation of path depths and AOCV derates. The AOCV retime is enabled with the `-retime aocv` or `aocv_path_slew_propagation` parameter. The AOCV stage counting mode (specified using the `timing_aocv_analysis_mode` attribute) is indicated in parenthesis. The stage counting modes are given below: • `launch_capture (default) (Default)` • `combine_launch_capture (Combine)` • `separate_clock_data (Separate)` • `clock_only (Clock)`
Corr	Indicates that the current SOCV accuracy mode considers correlation of variation between stages. Starting with SOCV accuracy mode `medium` (specified with the `delaycal_socv_accuracy_mode` attribute), it is assumed that some portion of the variation is not completely uncorrelated random variation, so the software tracks a correlated and uncorrelated component for each of the applicable moments.
Data Clock	Indicates the portion of the path that has been retimed. By default, only the data path is retimed. If the `timing_disable_retime_clock_path_slew_propagation` attribute is set to `false`, then the clock path will also be retimed, and “Clock” will be added to the retime status.
GP	Indicates that graph-based PBA mode settings are enabled.
GS	Indicates that GBA skewness mode settings are enabled.

Path-Slew	Indicates the selected retime analysis is set to use path correct slew rates to reduce the pessimism from slew merging. This can be controlled with the `report_timing -retime` parameter. The path slews are recalculated in `path_slew_propagation` and `aocv_path_slew_propagation` modes.
RC	Indicates that RC variation is enabled with the `timing_socv_rc_variation_mode` attribute setting, and a SOCV format RC variation side file library is available.
SI	Indicates that SI analysis was enabled during retime analysis. The path-based analysis will recompute SI effects using path correct victim timing windows.
SOCV(Ultra)	Indicates that SOCV analysis is enabled and has been incorporated into the retiming. The SOCV analysis mode (specified with the `delaycal_socv_accuracy_mode` attribute) is shown in parenthesis. The accuracy modes are mapped as follows: • `low` `<not shown>` • `medium` `Medium` • `high` `High` • `ultra` `Ultra`

Spatial (CS)	Indicates that spatial derating side files have been loaded and an explicit spatial derate factor will be calculated. This is not shown for AOCV 2D derate tables, where the spatial and depth derates are combined. The GBA spatial derate mode (specified with the `timing_spatial_derate_distance_mode` attribute) is shown in parenthesis with the possible values below: • `chip_size (CS)` • `bounding_box (BB)` The GBA spatial derate mode does not affect the path-based analysis bounding box calculation.
WP	Indicates that waveform propagation is enabled (using the the `delaycal_equivalent_waveform_model` attribute). The path-based retime will also use this delay calculation accuracy setting when recomputing delays and transitions for the path.

Examples

- The following command reports paths leading to the pin(s) specified by the *pin_list*:

```
report_timing -to u55/D

Path 1: MET Setup Check with Pin u55/CK
Endpoint:   u55/D (^) checked with leading edge of 'CLK_W_4'
Beginpoint: u99/Q (v) triggered by leading edge of 'CLK_W_4'
Path Groups: {CLK_W_4}
Other End Arrival Time            0.150
- Setup                           0.176
+ Phase Shift                     10.000
= Required Time                   9.974
- Arrival Time                    0.827
= Slack Time                      9.148
Clock Rise Edge                                0.000
+ Clock Network Latency (Prop)                 0.150
= Beginpoint Arrival Time                      0.150
------------------------------------------------------------------------
Load    Slew    Delay    Arrival  Cell   Arc           Pin
                         Time
------------------------------------------------------------------------
0.013   0.116   -       0.150     -      CK ^          u99/CK
0.027   0.086   0.195   0.345     DFF    CK ^ -> Q v   u99/Q ->
1.006   1.140   0.482   0.827     INV    A v -> Y ^    u4/Y
1.006   1.140   0.000   0.827     DFF    D ^           u55/D
------------------------------------------------------------------------
```

- The following command shows a report similar to the first with the addition of net arc information.

```
report_timing -to u55/D -net

Path 1: MET Setup Check with Pin u55/CK
Endpoint:   u55/D (^) checked with leading edge of 'CLK_W_4'
Beginpoint: u99/Q (v) triggered by leading edge of 'CLK_W_4'
Path Groups: {CLK_W_4}
Other End Arrival Time            0.150
- Setup                           0.176
+ Phase Shift                     10.000
= Required Time                   9.974
- Arrival Time                    0.827
= Slack Time                      9.148
Clock Rise Edge                                0.000
+ Clock Network Latency (Prop)                 0.150
```

```
= Beginpoint Arrival Time                  0.150
-------------------------------------------------------------------
Load    Slew    Delay    Arrival  Cell   Arc           Pin
                         Time
-------------------------------------------------------------------
0.013   0.116   -        0.150     -     CK ^          u99/CK
0.027   0.086   0.195    0.345     DFF   CK ^ -> Q v   u99/Q ->
0.027   0.086   0.000    0.345     INV   -             u4/A
1.006   1.140   0.482    0.827     INV   A v -> Y ^    u4/Y
1.006   1.140   0.000    0.827     DFF   -             u55/D
-------------------------------------------------------------------
```

- The `report_timing` command also accepts `-collection` as an argument. For example,

  ```
  report_timing [report_timing -collection]
  set rpt [report_timing -collection]
  report_timing $rpt
  ```

- The following command displays the worst late path in the design. The format of the report is similar to the first example: `report_timing`
- The following command displays the worst late path to each violating endpoint that has a slack less than -1.0: `report_timing -max_slack -1.0`
- The following command displays all the late paths that end at port `out[2]` and that have negative slack up to a maximum of 1000 worst paths. If there are more than 1000 paths, only the 1000 worst paths are reported: `report_timing -to out[2] -max_paths 1000 -max_slack 0.0`
- The following command reports the worst late path that starts at `in[0]` and ends at `out[1]`: `report_timing -from in[0] -to out[1]`
- The following command displays the three worst paths that start at `in[1]` and end at `out[3]`. With reconvergent fanout, more than one path may exist: `report_timing -from in[1] -to out[3] -max_paths 3`
- The following command displays the ten worst paths. Only the paths between the specified pins are enumerated.With reconvergent fanout, more than one path may exist: `report_timing -from i102/Z -to i123/A -max_paths 10`
- The following command enumerates the worst ten paths through the given two pins, starting at the beginning points in the design and ending at the endpoints, similar to using the `-from` option: `report_timing -through i102/Z -through i123/A -max_paths 10`
- The following command reports the ten worst paths through blocks A and C, or blocks B and C. The path only has to satisfy one element in a through list, but all through lists must be satisfied.

The example can be thought of as ((A or B) and C):

```
report_timing -through {A/*B/*} -through {C/*} -max_paths 10
```

- The following command forces a path through pin A and pin B. Use the syntax shown in the example. Do not use `-through {A B}`: `report_timing -through A -through B`
- The following command displays the clock path through the clock root to Launch flop:

```
report_timing -through aBC_bs/bs_mex1/r2_00_q1_reg_10_/CK -to .... -path_type
full_clock
```

- You will not see the path for the launch clock if you specify the following command because the report_timing command starts reporting from the CK pin:

```
report_timing -from aBC_bs/bs_mex1/r2_00_q1_reg_10_/CK
```

- The first command displays the path leading to the `CK` pin.
- The following command shows the `-unconstrained` option:

```
report_timing -unconstrained
Path 1:Endpoint: OUT4 (^) (unconstrained output)
Beginpoint: u55/Q (^) triggered by leading edge of 'CLK_W_4'
Arrival Time                    0.365
+ Clock Network Latency (Prop)      0.150
= Beginpoint Arrival Time           0.150
------------------------------------------------------------------
Load   Slew   Delay   Arrival Cell  Arc      Pin
                       Time
------------------------------------------------------------------
0.013   0.116   -       0.150    -      CK ^          u55/CK
0.011   0.072   0.215   0.365    DFF    CK ^ -> Q ^   u55/Q
0.011   0.131   0.000   0.365    -      OUT4 ^        OUT4 ->
------------------------------------------------------------------
```

- The following command sets a false path from u99/Q to u55/D:

```
set_false_path -from u99/Q -to u55/D
set_false_path -through u4/Y
report_timing -from u99/Q -to u55/D -unconstrained -path_exceptions all

  Path 1:Endpoint: u55/D (^) (unconstrained output)
  Beginpoint: u99/Q (v) triggered by leading edge of 'CLK_W_4'
  Arrival Time                  0.827
```

```
+ Clock Network Latency (Prop)      0.150
= Beginpoint Arrival Time           0.150
-------------------------------------------------------------------
Load    Slew    Delay   Arrival   Cell     Arc           Pin
                        Time
-------------------------------------------------------------------
0.013   0.116   -        0.150    -        CK ^          u99/CK
0.027   0.086   0.195    0.345    DFF      CK ^ -> Q v   u99/Q ->
1.006   1.140   0.482    0.827    INV      A v -> Y ^    u4/Y
1.006   1.140   0.000    0.827    DFF      D ^           u55/D ->
-------------------------------------------------------------------
Applied exceptions:
---------------------------------
Through            Late
---------------------------------
u4/Y               false
---------------------------------
Ignored exceptions:
-------------------------------------------------
From               To               Late
-------------------------------------------------
u99/Q              u55/D            false
-------------------------------------------------
```

- For hold paths, if the path has a setup multi-cycle path then those exceptions will also be reported because hold slacks get impacted by inferred path exceptions. For example:

```
set_multicycle_path 2 -setup -from IN1
```

```
Path 1: VIOLATED Hold Check with Pin seg1/u3/CK
Endpoint: seg1/u3/D (v) checked with leading edge of 'CLK_W_1'
Beginpoint: IN1     (v) triggered by leading edge of '@'
Path Groups: {CLK_W_1}
Other End Arrival Time        0.682
+ Hold                       -0.034
+ Phase Shift                 0.000
- Cycle Adjustment          -20.000
= Required Time              20.648
Arrival Time                  0.117
Slack Time                  -20.531
Clock Rise Edge                          0.000
+ Input Delay                            0.000
+ Drive Adjustment                      -0.014
= Beginpoint Arrival Time                -0.014
-------------------------------------------------------------------
```

```
Load    Slew    Delay   Arrival   Cell     Arc           Pin
                        Time
-------------------------------------------------------------------------
0.013   0.029   -       -0.014    -        IN1 v         IN1
0.013   0.057   0.054   0.040     CLKBUF   A v -> Y v    seg1/u1/Y
0.003   0.050   0.077   0.117     CLKBUF   A v -> Y v    seg1/u2/Y
0.003   0.050   0.000   0.117     DFF      D v           seg1/u3/D
-------------------------------------------------------------------------
Applied exceptions:
-----------------------------------------------
From               Early           Late
-----------------------------------------------
IN1                -               cycles 2
-----------------------------------------------
```

- The following command displays the timing report for analysis view `setup_view1`:

```
report_timing -view setup_view1
```

```
  Path 1: MET Late External Delay Assertion
  Endpoint:   EJ_TDOzstate      (^) checked with leading edge of 'virt_out_EJ_TCK'
  Beginpoint: core_ejt_reg_0_/Q (^) triggered by trailing edge of 'EJ_TCK'
  Path Groups: {virt_out_EJ_TCK}
  Analysis View: setup_view1
  Other End Arrival Time              0.000
  + Source Insertion Delay            0.300
  - External Delay                    1.250
  + Phase Shift                      17.400
  + CPPR Adjustment                   0.000
  = Required Time                    16.450
  - Arrival Time                      9.318
  = Slack Time                        7.132
  Clock Fall Edge                             8.700
  + Clock Network Latency (Ideal)             0.000
  = Beginpoint Arrival Time                   8.700
  ------------------------------------------------------------------------------
  Load   Slew   Delay   Arrival    Cell       Arc             Pin
                        Time
  ------------------------------------------------------------------------------
  0.010  0.000          8.700                 CP ^            core_ejt_reg_0_/CP
  0.118  0.145  0.127   8.828      DFSVTD4    CP ^ -> Q ^     core_ejt_reg_0_/Q
  0.525  0.438  0.411   9.238      BUFVTD6    I ^ -> Z ^      FE_EJ_TDOzstate/Z
  0.525  0.474  0.080   9.318                 EJ_TDOzstate ^  EJ_TDOzstate
  ------------------------------------------------------------------------------
```

- The following command shows a path slew propagation report:

```
report_timing -retime path_slew_propagation

Path 19: VIOLATED (-25.856 ns) Hold Check with Pin bidi_ff_in2/CK->D
Retime Analysis: Path-Slew
View: default_analysis_view_hold
Startpoint: (F) SEL
Clock:
Endpoint: (F) bidi_ff_in2/D
Clock: (R) clk_bidi

Capture Launch
Clock Edge:+ 0.000 0.000
Src Latency:+ 0.000 0.000
Net Latency:+ 41.379 (P) 0.000 (I)
Arrival:= 41.379 0.000

Hold:+ -0.031
Phase Shift:+ 0.000
Cppr Adjust:- 0.000
Required Time:= 41.348
Launch Clock:- 0.000
Data Path:- 15.493
Slack:= -25.856
+--------------------------------------------------------------------------------
----------------+
|        Pin      | Cell   | Arc        |Delay |Retime| Slew | Aocv | Aocv
|Arrival|Fanin|Fanout|
|                 |        |            |      | Delay|      |Derate| Stage|
Time  |     |     |
|                 |        |            |      |      |      |      | Count|
 |     |     |
|---------------------------------+----------+--------------+-------+--------+--
-----+--------+|
| SEL             |        |SEL v       |      |      |0.000 |0.824 |6.000 |
0.000 | 3   | 4   |
| sub/bd_dt/it/P  |PLBI2F  |            |0.000 |0.000 |0.000 |0.824 |6.000 |
0.000 | 3   | 4   |
| sub/bd_dt/it/D  |PLBI2F  |P v -> D v  |0.454 |0.409 |0.041 |0.822 |5.000 |
0.409 |     |     |
| sub/bd_in1/it/NN|PLBI2F  |            |0.000 |0.000 |0.041 |0.824 |6.000 |
0.409 | 1   | 3   |
| sub/bd_in1/it/P |PLBI2F  |NEN v -> P v|0.000 |0.000 |0.000 |0.822 |5.000 |
0.409 |     |     |
| bd_ptdt_in1/it/P|PLBI2F  |            |0.000 |0.000 |0.000 |0.824 |6.000 |
0.409 | 3   | 3   |
```

```
| bd_ptdt_in1/it/D|PLBI2F  |P v -> D v  |0.400 |0.360 |0.028 |0.818 |3.000 |
0.769 |     |     |
| bidi_buf_1/A    |BUFX2   |            |0.000 |0.000 |0.028 |0.820 |4.000 |
0.769 | 1   | 1   |
| bidi_buf_1/Y    |BUFX2   |A v -> Y v  |8.180 |7.362 |0.034 |0.818 |3.000 |
8.131 |     |     |
| bidi_buf_2/A    |CLKBUFX8|            |0.000 |0.000 |0.034 |0.820 |4.000 |
8.131 | 1   | 1   |
| bidi_buf_2/Y    |CLKBUFX8|A v -> Y v  |8.180 |7.362 |0.022 |0.818 |3.000
|15.493 |     |     |
| bidi_ff_in2/D   |DFFHQX1 |            |0.000 |0.000 |0.022 |0.820 |4.000
|15.493 | 1   | 1   |
+--------------------------------------------------------------------------------
--------------+
```

Sample Reports

Path Summary Report

```
report_timing -path_type summary
```

A sample path summary report is shown below:

```
#################################################################################
#
AAE DB initialization (MEM=750.941 CPU=0:00:00.2 REAL=0:00:00.0)
End delay calculation. (MEM=855.418 CPU=0:00:03.7 REAL=0:00:04.0)
Path 1: VIOLATED (-1.987 ns) Setup Check with Pin
DTMF_INST/TDSP_CORE_INST/EXECUTE_INST/sel_op_b_reg_0/CK->E
Group: vclk1
Startpoint: (R) reset
Clock: (R) vclk1
Endpoint: (F) DTMF_INST/TDSP_CORE_INST/EXECUTE_INST/sel_op_b_reg_0/E
Clock: (R) vclk1

Capture Launch
Clock Edge:+ 10.000 0.000
Src Latency:+ 3.000 4.000
Net Latency:+ 0.000 (I) 0.000 (I)
Arrival:= 13.000 4.000
```

```
Setup:- 0.956
Uncertainty:- 0.250
Required Time:= 11.794
Launch Clock:- 4.000
Input Delay:- 5.000
Data Path:- 4.781
Slack:= -1.987
```

Advanced Summary Section

The ‘advanced summary’ section shows specific information, such as, slack analysis, SI pessimism. This section provides cell-by-cell information on the path as well as nets, hierarchical pins, exceptions, and other aspects of the timing path.

A sample advanced summary report is shown below.

```
Path 1: MET (2285 ps) Setup Check with Pin rt/mul_73_17/retime_s5_16_reg/CK->D
. . .
Exceptions/Constraints:
path_adjust       370      scripts/constraints.sdc_line_57
input_delay      1001      scripts/constraints.sdc_line_27

Slack Analysis(3):
Stage(-3): (2123.60 ps) rt/mul_73_17/retime_s2_16_reg/D
Stage(-2): (2210.60 ps) rt/mul_73_17/retime_s3_16_reg/D
Stage(-1): ( 348.60 ps) rt/mul_73_17/retime_s4_16_reg/D
Stage( 0): (2285.30 ps) rt/mul_73_17/retime_s5_16_reg/D
Stage( 1): (-816.00 ps) rt/mul_73_17/retime_s6_9_reg/D
Stage( 2): (1283.90 ps) o6[15]
. . .
```

Note: These reports may contain content that has substantial runtime impact (such as meta-data, etc.).

Path Description Section

The path description section shows the cell-by-cell delay, slew, arc, exception information, and informational flag legends.

A sample path description section is shown below.

```
#-------------------------------------------------------------------------------
# Timing Point            Arc     Edge    Cell     Load     Trans   Delay  Arrival
#                                                  (fF)     (ps)    (ps)   (ps)
#-------------------------------------------------------------------------------
```

```
FF_I1_reg[0]/CK        <<< -      R       -          -      150     -       0
FF_I1_reg[0]/Q            CK->Q   R       DFFHQX4    45.4   201     334     334
mul_26_28/Fn0755D13176/Y  A->Y    F       INVX2      15.7   103     68      402
mul_26_28/Fn0755D13172/Y  A->Y    R       INVX4      53.0   223     130     532
mul_26_28/n0617DTtn0525/Y B->Y    F       NAND2X1    10.9   169     108     640
mul_26_28/n0634Dtn0548/Y  A1N->Y  F       AOI2BB2X4  19.3   164     232     872
mul_26_28/Fn0612D/Y       A->Y    R       INVX2      18.5   183     119     992
. . .
mul_26_43/n0014Dtn0331/Y  A->Y    F       NAND2X2     17.3   147    94      4168
mul_26_43/Fn0013D16832/Y  A->Y    R       INVX1       5.6    145    97      4265
mul_26_43/n0078Dtn0331/Y  A1N->Y  R       OAI2BB1X2   18.9   201    203     4468
mul_26_43/n0064Dtn0331/Y  A->Y    R       XOR2X4      10.6    86    225     4693
mul_26_43/n0025Dtn0331/S  CI->S   F       ADDFHX4     5.2     82    269     4962
FF_O2_reg[0]/D          <<< -     F       DFFHQX2     -       -       0     4962
#-------------------------------------------------------------------------------
```

By default, the timing reports show the ‘path summary’ and ‘path description’ section. However, the default can be changed by using the `report_timing -path_type` parameter.

End Point Reports

In addition to detailed reports, endpoint reports can also be generated. These reports allow you to include additional endpoint related information that can be useful in debugging or identifying certain issues. For example, an endpoint report can show the number of buffers and inverters on a path. A sample report is as shown:

```
> report_timing -path_type endpoint
#-----------------------------------
# ID   Slack    Endpoint         Cost
#      (ps)                      Group
#-----------------------------------
1     -5492     o4[2]             ck4
2     -5471     mult/data_reg[3] ck4
3     -5455     o4[1]             ck4
4     -5453     o4[0]             ck4
5     -5212     o3[1]             ck3
#-----------------------------------
```

The endpoint reports and detailed reports are mutually exclusive.

Clock Tracing Reports

Optionally, the report includes the clock tracing section that helps in debugging issues pertaining to clocks used by the startpoint and/or endpoint of a path. Clock tracing traces the source of the clock and the detailed path taken to reach the launch and/or capture elements' clock pin. A sample report is shown below.

```
> report_timing -view func1@wc_cworst -path_type full_clock
                -from ff1_reg -to ff3_reg
                -fields {-fanout}
Path 1: MET (405 ps) Setup Check with Pin ff3_reg/CK->D
          View: func1@wc_cworst
          Group: ck2
     Startpoint: (R) ff1_reg/CK
          Clock: (R) ck1
       Endpoint: (R) ff3_reg/D
          Clock: (R) ck2

                Capture        Launch
Clock Edge:+       1000             0
Src Latency:+         0             0
Net Latency:+         0 (I)         0 (I)
Arrival:=          1000             0

Setup:-              61
Required Time:=     939
Launch Clock:-        0
Data Path:-         595
Slack:=             405

#-------------------------------------------------------------------
# Timing Point   Arc   Edge  Cell   Load   Trans   Delay Arrival
#                                    (fF)   (ps)    (ps)   (ps)
#-------------------------------------------------------------------
clk1             -      R     (port)  0.9      0    0        0
clk1_buf1/Y      A->Y   R     BUFX4   0.9     24   60       60
clk1_buf2/Y      A->Y   R     BUFX4   0.0      0   67       127
ff1_reg/CK       -      R     DFFX1     -      0    0       127
#-------------------------------------------------------------------
ff1_reg/QN <<<   CK->QN R     DFFX1   0.6     23  335       462
ff1_i2_and/Y     B->Y   R     AND2X2  1.2     45  133       595
ff3_reg/D <<<    -      R     DFFX4     -      -    0       595
#-------------------------------------------------------------------
clk2             -      R    (port)   0.9      0    0      1000
clk2_buf1/Y      A->Y   R     BUFX4   2.8     31   64      1164
```

```
clk2_buf2/Y      A->Y   F     INVX4   2.8     22   24      1188
clk2_buf3/Y      A->Y   R     INVX4   0.0      0   14      1102
ff3_reg/CK       -      R     DFFX4   -        -    0      1102
#-------------------------------------------------------------
```

Scenarios for Group Path Reporting for User-Defined Groups

- A user-defined group is created with the same name as any of the defined reference clocks and with the same constraint as for the "to" clock. For example, `group_path -name CLK -to CLK`.

  In this case only one group named `CLK` will be displayed in the timing report. This will be the same default group with reference clock `CLK`. Note that default groups - clock gating and asynchronous checks - will not be honored for this group.

- A user-defined group is created with the same name as any of the reference clocks but with different constraint as specified for the "to" clock. E.g., `group_path -name CLK -from pin1 -to CLK`.

  In this case the two groups named `CLK` will be displayed in the timing report. One group will be the same as the default group with reference clock `CLK` (which will not honor clock gating or asynchronous checks) and the other will be the user-defined path group with name `CLK`, which will honor all the checks.

Related Information

- `report_nets`
- `report_ports`

report_timing_derate

```
report_timing_derate
[-delay_corner delayCornerName]
[-include_inherited]
[collection]
[{> | >>} filename[.gz]]
```

Reports delay scaling or derating factors for early and late clock and data paths in the current design.

If you specify this command without any parameters, reports are generated for global derating factors for each instance, net, or cell for which derating is asserted.

The following reports are also generated:

- In best case worst case, maximum and minimum corner
- Early values from minimum corner and late values from maximum corner in on-chip variation mode
- In MMMC mode, specified corners, or for all corners when corner is not specified
- Power domains (if present in the design)

Parameters

`> | >> filename`	When specified with `>`, writes the report to the specified file name. If the file already exists, the software overwrites it. When specified with `>>`, writes the report to the specified file name. If the file already exists, the software concatenates the report to the end of the file. **Note:** To write a compressed report, add the `.gz` extension to the file name.
`-delay_corner delayCornerName`	Reports derating factors for the specified delay calculation corner only. *Default:* Reports derating factors for all delay calculation corners.

`-include_inherited`	Reports the object names from which the derating factor values are inherited. When specified, the report displays the name of the cell for each instance from which derate values are inherited.
collection	Specifies the list of cells, nets, or library cells.

Examples

- The following commands display global derating factors:

```
set_timing_derate 1.2

set_timing_derate -clock -early 1.1

set_timing_derate -data -late 1.2

report_timing_derate
```

```
Output:
                 --------Clock-----------      ----------Data-----------------
                 Early              Late        Early             Late
                 ------------------------------------------------------------
Cell Delay       1.100000           1.200000    1.200000          1.200000
Net Delay Static 1.100000           1.200000    1.200000          1.200000
Net Delay Dynamic 1.100000          1.200000    1.200000          1.200000
Cell Check       --                 --          --                --
Retain delay     --                 --          --                --
```

- The following commands display instance derating factors:

```
set_timing_derate -cell_delay -clock -late 1.3 [get_cells {f2}]

set_timing_derate -cell_check -late 1.4 [get_cells {f2}]

report_timing_derate
```

```
Output:
                  --------Clock-----------      ----------Data-----------------
                  Early              Late        Early             Late
                  ------------------------------------------------------------
Cell Delay        --                 --          --                --
Net Delay Static  --                 --          --                --
Net Delay Dynamic --                 --          --                --
Cell Check        --                 --          --                --
Retain delay      --                 --          --                --


Cell (leaf): f2
Cell Delay        --              1.300000    --                --
Cell Check        --              1.400000    --                1.400000
```

- The following commands display hierarchical derating factors:

  ```
  set_timing_derate 1.2

  set_timing_derate 1.1 -late h4

  set_timing_derate 1.2 -early h4/h2

  set_timing_derate -cell_check 1.2 h4

  report_timing_derate -include_inherited
  ```

```
Output:
                  ---------------Clock-------------  -----------Data---------
                  Early                Late           Early             Late
                  ------------------------------------------------------------
Cell Delay        1.200000          1.200000       1.200000          1.200000
Net Delay Static  1.200000          1.200000       1.200000          1.200000
Net Delay Dynamic 1.200000          1.200000       1.200000          1.200000
```

```
Cell Check        --              --            --              --
Retain delay      --              --            --              --


Cell (hier): h4
Cell Delay        1.200000        1.100000      1.200000        1.100000
inherited         top             --            top             --
Net Delay Static  1.200000        1.100000      1.200000        1.100000
inherited         top             --            top             --
Net Delay Dynamic 1.200000        1.100000      1.200000        1.100000
inherited         top             --            top           --
Cell Check        1.200000        1.200000      1.200000        1.200000
inherited         --              --            --              --


Cell (hier): h4/h2
Cell Delay        1.200000        1.100000      1.200000        1.100000
inherited         --              h4            --            h4
Net Delay Static  1.200000        1.100000      1.200000        1.100000
inherited         --              h4            --              h4
Net Delay Dynamic 1.200000        1.100000      1.200000        1.100000
inherited         --              h4            --              h4
```

- The following commands display best case worst case derating factors:

```
set_timing_derate -cell_delay -clock -late 1.2 [get_cells {f2}] -min
set_timing_derate -cell_delay -late 1.3 [get_cells {f2}] -max
report_timing_derate
Output:
------------------------Report Timing Derate for MIN corner----------------------
                 ---------------Clock-------------  -----------Data---------
                 Early                Late           Early             Late
                 ------------------------------------------------------------
```

```
Cell Delay        --                 --              --                 --

Net Delay Static  --                 --              --                 --

Net Delay Dynamic --                 --              --                 --

Cell Check        --                 --              --                 --

Retain delay      --                 --              --                 --


Cell (leaf): f2

Cell Delay        --                 --              --                 1.200000


------------------------Report Timing Derate for MAX corner----------------------

                  ---------------Clock-------------  -----------Data---------

                  Early                Late           Early              Late

                  ------------------------------------------------------------

Cell Delay        --                 --              --                 --

Net Delay Static  --                 --              --                 --

Net Delay Dynamic --                 --              --                 --

Cell Check        --                 --              --                 --

Retain delay      --                 --              --                 --


Cell (leaf): f2

Cell Delay        --                 --              --                 1.300000
```

Related Information

- set_timing_derate

report_timing_summary

```
report_timing_summary
[-help]
[-checks {setup | hold | drv}]
[-expand_clocks {launch_capture | capture_only}]
[-expand_views]
[-groups groupname_list]
[-sort_by {name | wns | tns | fep}]
[-views viewName]
```

Generates reports that provide details summarized by the violation type - setup, hold, or DRV. You can sort the rows by WNS, TNS, FEP, or by view name.

Parameters

`-help`	Prints out the command usage.
`-checks {setup | hold | drv}`	Generates a timing report for the specified check types. The valid values are `setup`, `hold`, and `drv`. *Default*: `setup hold drv`
`-expand_clocks {launch_capture | capture_only}`	Includes per capture or launch/capture clock group details.
`-expand_views`	Includes per analysis view details.
`-groups` *`groupname_list`*	Generates a timing report for the specified path group(s). By default, all the path groups are analyzed.
`-sort_by {name | wns | tns | fep}`	Defines the order in which the report will be sorted - views, clocks, or path groups. The valid values are: • `name`: Places the items in alphabetical order. • `wns`: Sorts the output by WNS. • `tns`: Sorts the output by TNS. • `fep`: Sorts the output by FEP. *Default*: `name`
`-views` *`viewName`*	Generates a timing report for the specified analysis views only. By default all the active views are analyzed.

Examples

- The following example shows a sample `report_timing_summary` command output:

```
# SETUP NSIGMA WNS TNS FEP
#-------------------------------------------
View : ALL * -0.021 -18.719 5715
Group : Async * -0.003 -0.016 21
Group : ClockGate * -0.015 -0.920 393
Group : in2out * N/A N/A 0
Group : reg2out * N/A N/A 0
Group : in2reg * N/A N/A 0
Group : reg2reg * -0.021 -17.783 5301
#-------------------------------------------
# DRV NSIGMA WNS TNS FEP
#-------------------------------------------
View : ALL *
Check : max_transition * -0.028 -22.700 9356
Check : min_transition * N/A N/A 0
Check : max_capacitance * N/A N/A 0
Check : min_capacitance * N/A N/A 0
Check : max_fanout * N/A N/A 0
Check : min_fanout * N/A N/A 0
#-------------------------------------------
```

Related Commands

- `report_timing`

report_wire_load

```
report_wire_load
[-instances name_or_collection_of_hierarchical_instances]
[-ports name_or_collection_of_ports]
[-late | -early]
[> | >> filename]
```

Reports the wireload name, library, wire load mode and selection type for all hierarchical instances in the design.

Note: To enable wireload mode, you are required to set the following attribute:

set soceSupportWireLoadMode 1

Note: To use the wireload model functionality, you must enable the `enable_wire_load_model_support` attribute using the following command:

```
set_var enable_wire_load_model_support 1
```

Parameters

`>|>>` *`filename`*	When specified with `>`, writes the report to the specified file name. If the file already exists, the software overwrites it. When specified with `>>`, writes the report to the specified file name. If the file already exists, the software concatenates the report to the end of the file. The *`filename`* parameter must be the last argument in the list. **Note:** To write a compressed report, add the `.gz` extension to the file name. *Default:* Report is displayed on standard output without being saved.

`-instances` *`name_or_collection_of_hierarchical_instances`*	Specifies the name of an instance or collection for which the `report_wire_load` command reports the wireload name, library, wire load mode, and selection type. The `report_wire_load` command also supports wildcard names such as U*.
`-late	-early`	Reports wireload models in setup or hold analysis mode. By default, the software reports wireload models for the mode that you set using the command.
`-ports` *`name_or_collection_of_ports`*	Reports wireload information for the specified ports.

Examples

- The following example shows the default report when wireload mode is set to top:

```
report_wire_load
---------------------------------------------------------------------
Design Name top_chip
Wire Load Mode top

Instance Module WireLoadModel Library SelctionType
---------------------------------------------------------------------
- top_chip model4 fast library-default
---------------------------------------------------------------------
```

- The following example shows the report when wireload mode is set to enclosed to report lower hierarchy:

```
set_wire_load_mode enclosed

report_wire_load
---------------------------------------------------------------------
Design Name top_chip
Wire Load Mode enclosed
Instance Module WireLoadModel Library SelctionType
---------------------------------------------------------------------
```

```
- top_chip model4 fast library-default
U1 randr model4 fast library-default
U2 randr model4 fast library-default
------------------------------------------------------------------------
```

- The following example shows model assertions using the `set_wire_load_model` command the corresponding reports using the `report_wire_load` command:

```
set_wire_load_model -name model3
report_wire_load
------------------------------------------------------------------------
Design Name top_chip
Wire Load Mode enclosed

Instance Module WireLoadModel Library SelctionType
------------------------------------------------------------------------
- top_chip model3 fast user-asserted
U1 randr model3 fast user-asserted
U2 randr model3 fast user-asserted
------------------------------------------------------------------------
set_wire_load_model -name model2 [get_cells U1]

set_wire_load_model -name model4 [get_cells U2]

report_wire_load
------------------------------------------------------------------------
Design Name top_chip
Wire Load Mode enclosed

Instance Module WireLoadModel Library SelctionType
------------------------------------------------------------------------
- top_chip model3 fast user-asserted
U1 randr model2 fast user-asserted
U2 randr model4 fast user-asserted
------------------------------------------------------------------------
```

- The following example shows the report including port information:

```
report_wire_load -ports [get_ports XYZ/A]

-----------------------------------------------------------------------
Design Name   randr

Wire Load Mode   enclosed

Port     Module WireLoadModel Library    SelctionType
-----------------------------------------------------------------------

P1     randr model2               slow   user-asserted

P2     randr model4              slow   user-asserted
-----------------------------------------------------------------------
```

Related Commands

- set_wire_load_model
- reset_wire_load_model

reset_instance_library

```
reset_instance_library
[-help]
```

Resets the settings/binding done using the `set_instance_library` command.

Parameters

`-help`	Prints out the command usage.

Related Commands

- `set_instance_library`

reset_path_adjust

```
reset_path_adjust
[-help]
-path_adjust_group <path_adjust_group_name>
-view <string>
[-setup | -hold]
```

Removes the specified slack adjustment values for timing paths that are set using the set_path_adjust command.

Parameters

-help	Prints out the command usage.
-hold	Resets the adjust value for early paths.
-path_adjust_group <path_adjust_group_name>	Specifies the name of path adjust group that defines the path for which the adjustment will be applied.
-setup	Resets the adjust value for late paths.
-view <string>	Specifies the view for which adjustment will be reset.

Related Commands

- set_path_adjust

reset_path_adjust_group

```
reset_path_adjust_group
[-help]
[-name <group_name> | -all ]
```

Removes all path adjust groups that are set using the `set_path_adjust_group` command for setting path slack adjustment values.

Parameters

`-help`	Prints out the command usage.
`-all`	Removes all the path adjust groups.
`-name` *`group_name`*	Removes the named path adjust group completely.

Related Commands

- `set_path_adjust_group`
- `set_path_adjust`

reset_sdf_assertions

`reset_sdf_assertions`

Resets the sdf delays to the values defined in the library. When you perform timing analysis using the timing engine, the software uses sdf values that you define using the `read_sdf` command earlier in the flow. You use the `reset_sdf_assertions` commands to discard delay values specified using the sdf files, and use the delays defined in the library.

Note: You do not need to read in the sdf delays every time you perform delay calculation. The software uses the sdf files specified earlier in the flow.

Parameter

None

Related Commands

- `read_sdf`

reset_timing_derate

```
reset_timing_derate
[-help]
[-delay_corner delayCornerName]
[-increment]
[obj_list]
```

Resets derate factors specified on a design or a list of instances (cells, net, or library cells). All the derating factors that are set globally or on specific instances are reset to a default value of `1.0`.

Derating factors for early or late paths in the design are set using the `set_timing_derate` command.

Parameters

`-help`	Prints out the command usage.
`-delay_corner delayCornerName`	Specifies the delay corner or a list of delay corners on which reset derating is to be applied.
`-increment`	Incrementally adds the derate value to the total derate value in all the modes - OCV, AOCV, and SOCV mode.
obj_list	Specifies the list of cells, nets, or library cells to reset. If the *obj_list* contains hierarchical cells then all cells within the hierarchical cells and across lower hierarchies have reset derate factors.

Examples

- The following example resets all the object-specific and global derating factors:
  ```
  reset_timing_derate *
  ```
- The following example resets only global derating factors:
  ```
  reset_timing_derate [current_design]
  ```

- The following example resets derating factor values for the specified delay corner:
  `reset_timing_derate -delay_corner` *`delayCornerName`*
- The following example resets the global derating values for the specified delay corner:
  `reset_timing_derate -delay_corner` *`delayCornerName`* `[current_design]`
- The following example resets timing derate on instance `u1`:
  `reset_timing_derate [get_cells u1]`

Related Commands

- `set_timing_derate`

reset_wire_load_mode

`reset_wire_load_mode`

Resets the wire-load mode to the default mode of `top`.

Note: To enable wireload mode, you are required to set the following attribute:

```
set soceSupportWireLoadMode 1
```

Note: To use the wireload model functionality, you must enable the `enable_wire_load_model_support` attribute using the following command:

```
set_var enable_wire_load_model_support 1
```

Related Information

`set_wire_load_mode`

reset_wire_load_model

```
reset_wire_load_model
[-help]
[{ obj_list }]
```

Resets wireload models for the specified list of objects. The list of objects can include designs, hierarchical cells, or ports.

When objects are not specified, the software removes the wireload model specified using the `set_wire_load_model` command from the current instance only.

Note: To enable wireload mode, you are required to set the following attribute:

```
set soceSupportWireLoadMode 1
```

Note: To use the wireload model functionality, you must enable the `enable_wire_load_model_support` attribute using the following command:

```
set_var enable_wire_load_model_support 1
```

Parameters

`-help`	Outputs a brief description that includes type and default information for each `reset_wire_load_model` parameter. For a detailed description of the command and all of its parameters, use the man command: `man reset_wire_load_model`.
`obj_list`	Specifies the list of instances or ports for which to remove the wireload models.

Example

- The following command removes the wireload model for `clk1`:

  ```
  reset_wire_load_model clk1
  ```

Related Commands

- set_wire_load_model

reset_wire_load_selection_group

```
reset_wire_load_selection_group [{ obj_list }]
```

Removes the wireload selection group for the specified list of objects. The list of objects can include designs, hierarchical cells, or ports. When objects are not specified, wireload selection groups are removed for all objects.

Note: To enable wireload mode, you are required to set the following attribute:

```
set soceSupportWireLoadMode 1
```

Note: To use the wireload model functionality, you must enable the `enable_wire_load_model_support` attribute using the following command:

```
set_var enable_wire_load_model_support 1
```

Parameters

obj_list	Specifies the list of object names for which to remove the wireload selection groups.

Example

- The following command removes the wireload selection group for `clk1`:

  ```
  reset_wire_load_selection_group clk1
  ```

Related Commands

- set_wire_load_selection_group

set_analysis_view

```
set_analysis_view
-help
[-drv string]
-dynamic string
-hold list_of_views
-leakage string
-setup list_of_views
-update_timing
```

Defines the analysis views to use for setup and hold analysis and optimization.You must define at least one setup and one hold analysis view. The "active" views represent the different design variations that will be analyzed. These views can be changed throughout the flow to utilize different subsets of views. The libraries and data are loaded into the system that are required to support the selected set of active views.

When the `set_analysis_view` command is issued, it will cause a full timing reset. During a timing reset, all the interconnect parasitics, delays, and timing slacks will be recalculated.

The order in which you specify the views with the `-setup` and `-hold` parameters is important. By default, the first views defined in the `-setup` and `-hold` lists are the default views. The default views serve two purposes:

- Provide the library data that is used during the design initialization process.
- Indicate to parts of the application, that are not MMMC-aware, which view's information to use for a particular analysis step (in most cases, the default view is the first setup or hold view). You can use the `set_default_view` command to change the default view to any currently active view without incurring a timing reset. This control is not frequently used since most applications either support MMMC or provide a direct control for selecting the appropriate view to use for that specific analysis.

Using the `set_default_view` command does not affect software performance because it only uses views that are already active in the design. If you use the `set_analysis_view` command to change the default views, the existing timing and delay calculation data is reset.

Use this command after creating multiple analysis views (`create_analysis_view`).

Note: The `set_analysis_view` command does not work in an incremental manner for setting parameters. Each invocation of this command must include the full specification of all the active views required for analysis.

Parameters

`-help`	Prints out the command usage.
`-drv` *string*	Allows you to set DRV views.
`-dynamic` *string*	Sets a view as dynamic view. Delay calculation is required to provide transition times. Delay calculation should not perform SI analysis if the delay corner is only required for the dynamic power. The timer is not required to propagate required times or perform slack computations.
`-hold` *list_of_views*	Sets the active views for hold analysis.
`-leakage` *string*	Sets a view as leakage view. The Timing libraries must be loaded, if not already in use. The Timer constant propagation is required, whereas delay calculation is not required.
`-setup` *list_of_views*	Sets the active views for setup analysis.

`-update_timing`	Reloads the configuration described by the MMMC view and mode objects. When the timing attribute `timing_defer_mmmc_obj_updates` is set to `true`, this parameter is required to update the MMMC timing environment. The `-update_timing` parameter may be specified without any `-setup` or `-hold` arguments. In this case, the data associated with the current set of setup and/or hold views will be loaded. If no active setup or hold views exist, then using this parameter alone may display an error. The `-update_timing` parameter may also be used in conjunction with the `-setup` and `-hold` options. In this case, the newly specified setup/hold view configuration will be loaded with the updated MMMC object information.

When you issue the `set_analysis_view` command, the software will perform a timing reset, so that only the new active view information is loaded up. The software requires the `-setup` and `-hold` view parameters to be explicitly specified for each `set_analysis_view` command run. If these parameters are not specified, the software issues an error message. Any optional settings, such as, `-leakage` or `-dynamic` views will be reset if they are not specified with the new `set_analysis_view` command.

Note: It is recommended to use the same power view for both leakage and dynamic, that is, both the `-dynamic` and `-leakage` parameters should have the same user-specified view settings.

Examples

- The following command sets `missionWCCOM` and `mission2WCCOM` as the active views for setup analysis, and `missionBCCOM` and `testBCCOM` as the active views for hold analysis:

  ```
  set_analysis_view -setup {missionWCCOM mission2WCCOM} -hold {missionBCCOM testBCCOM}
  ```

Related Topics

- Performing Multi-Mode Multi-Corner Timing Analysis and Optimization chapter in the *Innovus User Guide*

Related Commands

- create_analysis_view
- update_analysis_view

set_aocv_thresholds

```
set_aocv_thresholds
[-help]
[-min_stage_count_setup <int>]
[-min_stage_count_hold <int>]
[-reset]
[-slack_pruning_threshold <float>]
-view <list_of_views>
```

Limits the minimum AOCV stage depth in graph-based analysis (GBA) mode.

Parameters

`-help`	Prints out the command usage
`-min_stage_count_setup <int>`	Applies the AOCV stage count floor, during Setup analysis, so that all the instances are guaranteed to have a stage depth no less than the specified value. A value of '0' will remove the assertion.
`-min_stage_count_hold <int>`	Applies the AOCV stage count floor, during Hold analysis, so that all the instances are guaranteed to have a stage depth no less than the specified value. A value of '0' will remove the assertion.
`-reset`	Removes/sets to default all of the AOCV threshold assertions for specified views.
`-slack_pruning_threshold <float>`	Controls the slack threshold (in two-pass/slack-based AOCV analysis mode), at which paths are removed from consideration for the next phase of AOCV stage counting.
`-view <list_of_views>`	Applies the specified thresholds to the list of analysis views.

set_default_view

```
set_default_view
{-setup newDefaultSetupView | -hold newDefaultHoldView}
```

Temporarily changes the default active analysis view to a different active view.

Some Innovus applications can function only on a single analysis view at a time. By default, these single-view applications use the default analysis view. If an application or flow step does not provide a native option for specifying which view to use, the `set_default_view` command can be used to temporarily change the default view to a different active view that is better suited to that flow step.

Note: Using the `set_default_view` command does not affect software performance because it only uses views that are already active in the design. If you use the set_analysis_view command to change the default views, the existing timing, delay calculation, and RC data is reset.

Use this command after defining active views for the design (set_analysis_view).

Parameters

`-hold newDefaultHoldView`	Changes the default active hold analysis view to the specified analysis view.
`-setup newDefaultSetupView`	Changes the default setup analysis view to the specified analysis view.

Examples

- The following command sets `missionWCCOM`, `mission2WCCOM`, and `missionSlow` as the active views for setup analysis, and `missionBCCOM` and `testBCCOM` as the active views for hold analysis. The default active setup view is `missionWCCOM`, and the default active hold view is `missionBCCOM`.
  ```
  set_analysis_view -setup {missionWCCOM mission2WCCOM missionSlow} -hold {missionBCCOM testBCCOM}
  ```
- The following command changes the default active setup view to `missionSlow`:
  ```
  set_default_view -setup missionSlow
  ```
- The following command changes the default active hold view to `testBCCOM`:
  ```
  set_default_view -hold testBCCOM
  ```

Related Commands

- create_analysis_view
- set_default_view

set_instance_library

```
set_instance_library
[-help]
-library string
-instance string
[-early | -late]
```

Binds a cell library to an instance. When specified, the software overrides the default binding mechanism for a cell from the library set.

Note: The `set_instance_library` command cannot be used at the command prompt - this command can be specified in the ViewDefinition file.

Parameters

`-help`	Prints out the command usage.
`-early`	Uses the library binding for early timing analysis.
`-instance <string>`	Specifies the instance for which the library binding needs to be done.
`-late`	Uses the library binding for late timing analysis.
`-library <string>`	Specifies the library file path.

Examples

The following commands specify the instance for which the library binding will be done:

- `set_instance_library -library newcell.lib -instance {i1}`
- `set_instance_library -library mysil.lib -instance {i1 i2}`
- `set_instance_library -library {new.lib mysil.lib} -instance {i1} -early`
- `set_instance_library -library {mynew.lib mysil.lib} -instance {i1 i2} -late`

Related Commands

- `reset_instance_library`

set_interactive_constraint_modes

```
set_interactive_constraint_modes
{constraint_mode_list}
```

Puts the software into interactive constraint entry mode for the specified multi-mode multi-corner constraint mode objects. Any timing constraints that you specify after this command take effect immediately on all active analysis views that are associated with the specified constraint modes. By default, no constraint modes are considered active.

The software stays in interactive mode until you exit by specifying the `set_interactive_constraint_modes` command with an empty list:

```
set_interactive_constraint_modes { }
```

All new assertions are saved in the SDC files of the specified constraint modes when you save the design (`write_db`).

Use the `get_interactive_constraint_modes` command to return a list of the constraint mode objects in interactive constraint entry mode.

Note: Interactive constraint mode only works when the software is in multi-mode multi-corner timing analysis mode. In min/max analysis mode, constraints are always accepted interactively.

Parameters

{*constraint_mode_list*}	Specifies the constraint mode objects to update with the new assertions.

Examples

- The following commands put the software into interactive constraint entry mode, and apply the `set_propagated_clock` assertion on all views in the current session that are associated with the constraint mode `func1`:

  ```
  set_interactive_constraint_modes func1
  set_propagated_clock [all_clocks]
  ```

  When you specify `write_db`, the updated constraints file for the `func1` constraint mode will include the `set_propagated_clock`.

Related Commands

- get_interactive_constraint_modes
- get_db
- write_db

set_io_thresholds

```
set_io_thresholds
[-slew_lower_threshold_pct pctValue]
[-slew_upper_threshold_pct pctValue]
[-input_threshold_pct pctValue]
[-output_threshold_pct pctValue]
[-slew_lower_threshold_pct_rise pctValue]
[-slew_lower_threshold_pct_fall pctValue]
[-slew_upper_threshold_pct_rise pctValue]
[-slew_upper_threshold_pct_fall pctValue]
[-input_threshold_pct_rise pctValue]
[-input_threshold_pct_fall pctValue]
[-output_threshold_pct_rise pctValue]
[-output_threshold_pct_fall pctValue]
```

Sets the threshold trip-points used for delay calculation to and from the primary I/Os. The `set_io_thresholds` command does not affect the interpretation of threshold settings in the Liberty timing libraries; the effect is strictly limited to the I/O ports.

If you do not specify I/O thresholds with the `set_io_thresholds` command, delay calculation assumes that the I/O thresholds are equivalent to those of the logic gates to which they are connected.

The `set_io_thresholds` command supports separate rise and fall slew thresholds as normally found in Liberty timing libraries.

Note: Some third-party applications derive the I/O threshold values from the first Liberty timing library in the library search path. The software currently does not support this methodology. Explicit setting of I/O thresholds with the `set_io_thresholds` command should be used to complete a compatible analysis environment.

Parameters

`-input_threshold_pct` *pctValue*	Specifies the slew input threshold value.
`-input_threshold_pct_fall` *pctValue*	Specifies the measurement point for a falling signal that will be used for calculating the interconnect delay from a primary input to internal logic. *Value:* `0` - `100` *Default:* `50`

`-input_threshold_pct_rise` *pctValue*	Specifies the measurement point for a rising signal that will be used for calculating the interconnect delay from a primary input to internal logic. *Value:* `0 - 100` *Default:* `50`
`-output_threshold_pct` *pctValue*	Specifies the slew output threshold value.
`-output_threshold_pct_fall` *pctValue*	Specifies the measurement point for a falling signal that will be used for calculating the interconnect delay from internal logic to a primary output. *Value:* `0 - 100` *Default:* `50`
`-output_threshold_pct_rise` *pctValue*	Specifies the measurement point for a rising signal that will be used for calculating the interconnect delay from internal logic to a primary output. *Value:* `0 - 100` *Default:* `50`
`-slew_lower_threshold_pct` *pctValue*	Specifies the slew lower threshold value.
`-slew_lower_threshold_pct_fall` *pctValue*	Specifies the lower trip-point for a falling signal that will be used to measure transition times. *Value:* `0 - 100` *Default:* `20`

`-slew_lower_threshold_pct_rise` *pctValue*	Specifies the lower trip-point for a rising signal that will be used to measure transition times. *Value:* `0 - 100` *Default:* `20`
`-slew_upper_threshold_pct_fall` *pctValue*	Specifies the upper trip-point for a falling signal that will be used to measure transition times. *Value:* `0 - 100` *Default:* `80`
`-slew_upper_threshold_pct_rise` *pctValue*	Specifies the upper trip-point for a rising signal that will be used to measure transition times. *Value:* `0 - 100` *Default:* `80`
`-slew_upper_threshold_pct` *pctValue*	Specifies the slew lower rise threshold value.

Examples

The following command specifies trip-points for measuring delay and transition times:

```
set_io_thresholds -slew_lower_threshold_pct_rise 10


-slew_upper_threshold_pct_fall 80
-slew_upper_threshold_pct_rise 90
-slew_lower_threshold_pct_fall 20
-input_threshold_pct_rise 40
-input_threshold_pct_fall 70
-output_threshold_pct_rise 60
-output_threshold_pct_fall 30
```

set_path_adjust

```
set_path_adjust
[-help]
adjust_value
[-path_adjust_group path_adjust_group_name]
[-view string]
[-setup | -hold]
```

Defines the slack adjustment value for timing paths. The defined adjustment value is added to the slack of a path specified by the path adjust group, referred to by the `-path_adjust_group` parameter. You can specify a path adjust group by using the `set_path_adjust_group` command.

If multiple path adjustment values are applicable on the same pin, which is on the path specified by multiple `set_path_adjust_group` constraints, then the smallest adjustment value will be applied.

Note: The `set_path_adjust` command is not allowed to be used inside the SDC constraints file. It is only allowed in the software run script.

Parameters

`-help`	Prints out the command usage.
adjust_value	Specifies the value of slack adjustment.
`-hold`	Specifies to apply the adjust value to early paths.
`-path_adjust_group` *path_adjust_group_name*	Specifies the name of path adjust group that defines the path for which the adjustment will be applied.
`-setup`	Specifies to apply the adjust value to late paths. **Note**: If you do not specify `-setup` or `-hold` parameter, the software sets the adjust value to both setup and hold analysis.
`-view` *string*	Specifies a list of analysis views for which adjustment will be applied. This parameter accepts multiple views as argument.

Examples

- The following commands define the slack adjustment values for the specified views:

```
> set_path_adjust 0.44 -path_adjust_group AG1 -view view1 -hold
> set_path_adjust 0.44 -path_adjust_group AG2 -view view2 -setup
> set_path_adjust 0.44 -path_adjust_group AG3 -view view3
```

Related Commands

- set_path_adjust_group

set_path_adjust_group

```
set_path_adjust_group
[-help]
-name <path_adjust_group_name>
[-from <from_list>]
[-to <to_list>]
[-through <through_list>]
```

Allows you to specify the path that is used by the `set_path_adjust` command for setting path slack adjustment values.

The `set_path_adjust_group` command is not allowed to be used inside the SDC constraints file. It is only allowed in the tool run script.

If path group definitions are matching/overlapping (when the `-from`/`-through`/`-to` parameters are used), then path group resetting is applied. The path group overwriting and prioritization rules are similar to path exceptions.

For more details, refer to `group_path`.

Parameters

`-help`	Prints out the command usage.
`-from <from_list>`	Specifies a list of pins, instances, or clocks that are the start of the paths.
`-name <path_adjust_group_name>`	Specifies the name of the path adjust group.
`-through <through_list>`	Specifies a list of pins or instances where the paths go through.
`-to <to_list>`	Specifies a list of pins, instances, or clocks where the path ends.

Examples

The following commands specify the adjusted path groups:

- `set_path_adjust_group -name AG1 -from in1 -through buf1/A -to out1`
- `set_path_adjust_group -name AG2 -from [get_clocks CK1] -through buf1/A -to [get_clocks CK2]`

- `set_path_adjust_group -name AG3 -from [get_clocks {CK1 CK2}] -to [get_clocks {CK1 CK2}]`

Related Commands

- `set_path_adjust`
- `reset_path_adjust_group`
- `group_path`

set_socv_rc_variation_factor

```
set_socv_rc_variation_factor
[-help]
variation_factor
[-early]
[-late]
[-views list_of_views]
```

Defines the value of interconnect delay variation factor.

This command applies global variation as percentage of the nominal delay.

To enable this feature, you can set the `timing_socv_rc_variation_mode` attribute to `true`.

Parameters

`-help`	Prints out the command usage
`variation_factor`	Specifies the value of the interconnect delay variation factor to be used.
`-early`	Specifies the variation factor for early interconnect delays.
`-late`	Specifies the variation factor for late interconnect delays.
`-views` *`list_of_views`*	Apply the SOCV interconnect variation factor to the specified list of analysis views.

set_socv_reporting_nsigma_multiplier

```
set_socv_reporting_nsigma_multiplier
[-help]
[-views list_of_views]
{[-setup float] [-hold float]}
[-transition float]
```

Controls n-sigma value for reporting purposes.

This command supports n-sigma specification per view per setup/hold combination.

Parameters

`-help`	Prints out the command usage.
`-hold float`	Specifies the sigma multiplier for hold mode.
`-setup float`	Specifies the sigma multiplier for setup mode.
`-transition float`	Specifies the sigma multiplier to use for transition related DRV checks – including max_transition, min_transition, pulse_clock_max_transition, and pulse_clock_min_transition. When this parameter is not specified, the sigma multiplier specified for setup will be used for transitions. When the `-transition` parameter is specified, you can set the `timing_report_max_transition_check_using_nsigma_slew` attribute to `true` to report transition related DRV checks, such as, max_transition, min_transition, pulse_clock_max_transition, and pulse_clock_min_transition. This setting enables the `report_constraint` command to use mean and sigma slew values for reporting transition values in DRV checks.
`-views list_of_views`	Applies the sigma multiplier factor to the specified list of analysis views. **Note**: The specified views should be active.

Examples

- The following command applies the sigma multiplier to the specified view in setup mode :
  ```
  set_socv_reporting_nsigma_multiplier -setup 2.5 -views std_max_setup
  ```
- The following command applies the sigma multiplier to the specified view in for hold mode:
  ```
  set_socv_reporting_nsigma_multiplier -hold 3.0 -views std_min_hold
  ```

Related Commands

- get_socv_reporting_nsigma_multiplier

set_table_style

```
set_table_style
-help
-name string
[-reverse_rows]
[-major_sort integer]
[-minor_sort integer]
[-max_widths integer]
[-min_widths integer]
[-indent integer]
-frame | -no_frame_fix_width [-nosplit]
```

Disables printing of specific columns and allows for specifying the minimum and maximum size of each column in a report. It allows for reversing the data, for controlling the left indent, and for controlling the sorting of the columns.

Note: Do *not* include a blank space between commas ", " when specifying widths. If you are specifying a value for the width(s), make sure there is *no* leading or trailing blank space.

The `-frame`, `-no_frame`, and `-no_frame_fix_width` parameters work in a global manner. If you specify one of these parameters, it will affect all reports that can be formatted with this command. You cannot specify them with the -name parameter, to change the format of a specific report.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `set_table_style` parameter. For a detailed description of the command and all of its parameters, use the man command: `man set_table_style`.
`-frame`	Specifies that the table columns have a fixed frame or border, based on the minimum and maximum width settings. Any data that is longer than the maximum width for its column is wrapped within the same column. This is the default format for tables.
`-indent integer`	Specifies the number of spaces to leave on the left for that particular table.

`-major_sort` *integer*	Specifies the column to be used in the major sort. If the column number is negative, it reverses the sort.

`-max_widths` *integer*	Specifies the maximum width for each table column in the report specified by the `-name` parameter. **Note**: You cannot specify `-max_widths` parameter without `-name`. You can separate each specified width value with a comma (,). For example, if you specify: -max_widths `{10,20,30}` The first column will have a maximum width of 10, the second column will have a maximum width of 20, and the third column will have a maximum width of 30. The fourth and fifth columns will remain the default width spacing. Commas are also specified as column "placeholders", when you want to change the spacing of a column without changing the spacing of the preceding column. For example, if you specify: -max_widths {,,20,30} The third column will have a maximum width of 20, the fourth column will have a maximum width of 30, and the first, second, and fifth columns will remain the default width spacing. If you specify: `-max_widths {20,,30}` The first column will have a maximum width of 20, the third column will have a maximum width of 30, and the second, fourth and fifth columns will remain the default width spacing. Each column resizes itself to be between the range of the maximum and minimum widths specified for that column, based on the data in that column. Overruns are wrapped within the same column, unless otherwise specified. *Default:* `{50,50,50,50,50}`

`-min_widths` *integer*	Specifies the minimum width for each table column in the report specified by the `-name` parameter. **Note**: You cannot specify `-min_widths` parameter without `-name`. You can separate each specified width value with a comma (,). For example, if you specify: -min_widths `{7,10,15}` The first column will have a minimum width of 7, the second column will have a minimum width of 10, and the third column will have a minimum width of 15. The fourth and fifth columns will remain the default width spacing. Commas are also specified as column "placeholders", when you want to change the spacing of a column without changing the spacing of the preceding column. For example, if you specify: -min_width {,,10,15} The third column will have a minimum width of 10, the fourth column will have a minimum width of 15, and the first, second, and fifth columns will remain the default width spacing. If you specify: -min_width {7,,10} The first column will have a minimum width of 7, the third column will have a width of 10, and the second, fourth and fifth columns will remain the default width spacing. Each column resizes itself to be between the range of the maximum and minimum widths specified for that column, based on the data in that column. Overruns are wrapped within the same column, unless otherwise specified. *Default:* `{5,5,5,5,5}`

`-minor_sort` *integer*	Specifies the column to be used in the minor sort. It applies to *all* rows that have the same values in the `major_sort` column. If the column number is negative, the software reverses the sort.
`-name` *name*	Changes the formatting for the specified table. See Table for a list of table names you can use, and the commands with which they work.
`-no_frame_fix_width`	Specifies that the table columns have a fixed minimum width, but no maximum width (the data does not wrap). Column data that fits within the column's minimum width will left-justify and pad its location to fill the specified column space. Column data that is longer than the column's minimum width will take as much space in the table as needed. If a column is overwritten by the data from the preceding column, its data will begin on a new line at the correct column location. *Default:* `-frame`
`-nosplit`	Specifies that each column follows the preceding column without splitting to a new line, if the data in one column overruns the next column. The wrapping behavior for data is controlled by the size of the display window. This format is useful for machine parsing the report. **Note:** You must specify `-no_frame_fix_width` in order to use this parameter.
`-reverse_rows`	Reverses the data rows. This is useful, for example, if you want to trace back from an end point and the default report traces forward from the end point.

The timing report formatting depends on the specified `set_table_style` parameters. These are described below:

- When `set_table_style -no_frame_fix_width` parameter is specified, the timing report displays long inst/hpin names and the rest of the column values are shown on the next line.

- When `set_table_style -no_frame_fix_width -nosplit` is specified, the timing report displays long inst/hpin names and rest of the column values are shown on the same line, because `-nosplit` table style is specified. In this case, other column headers (beyond inst/pin) may not be aligned properly.

- The following settings are specified:

  ```
  set_table_style -no_frame_fix_width -nosplit
  set_table_style -name report_timing -max_width {50,50,,,,,}
  ```

  The generated report will show properly aligned column headers and end-point/begin-points are not split.
- The following settings generate a report displaying longer (than above reports) inst/pin names:

  ```
  set_table_style -no_frame_fix_width -nosplit
  set_table_style -name report_timing -max_width {120,120,,,,,}
  ```

  In this report output, the column headers are aligned properly and are not split.

Table: **List of Table Names**

Table Name	Formats Reports Generated by
`report_cell_instance_delays_arcs`	`report_cell_instance_timing`
`report_cell_instance_delays_checks`	`report_cell_instance_timing`
`report_cell_instance_delays_pins`	`report_cell_instance_timing`
`report_inactive_arcs`	`report_inactive_arcs`
`report_net`	`report_net`
`report_net_header`	`report_net`
`report_net_sink_table`	`report_net`
`report_net_source_table`	`report_net`
`report_net_summary_table`	`report_net`
`report_path_exceptions`	`report_path_exceptions`

report_port	report_ports
report_timing	report_timing
report_timing_summary	report_timing

Examples

- The following command resizes the first column to a maximum range of 70 and specifies a default width of 50 for the second column. The default for the maximum width is 50 and the default for the minimum width is 5:
  ```
  set_table_style -name report_timing -max_widths {70,}
  ```
  Note: Do *not* include a blank space between commas ",," when specifying widths. If you are specifying a value for the width(s), make sure there is *no* leading or trailing blank space.
- The following command specifies a default width of 50 for the first and second column:
  set_table_style -name report_timing -max_widths {,}

  The following sample timing report shows the table style generated with the `-frame` parameter:

  ```
  set_table_style -frame

  report_timing -format {cell arc delay arrival required instance}

  Path 1: MET Setup Check with Pin seg3/u14/CK
  Endpoint: seg3/u14/D (^) checked with leading edge of 'CLK_W_3'
  Beginpoint: seg3/u9/Q (^) triggered by leading edge of 'CLK_W_3'
  Path Groups: {CLK_W_3}
  Other End Arrival Time 0.621
  - Setup 0.186
  + Phase Shift 2.000
  = Required Time 2.434
  - Arrival Time 2.062
  = Slack Time 0.372
  Clock Rise Edge 0.000
  + Clock Network Latency (Prop) 0.682
  = Beginpoint Arrival Time 0.682
  +-------------------------------------------------------------------------+
  | Cell    | Arc         | Delay    | Arrival  | Required  | Instance  |
  |         |             |          | Time     | Time      |           |
  |------+------------+-------+---------+----------+-------------------|
  ```

```
|         | CK ^          |          | 0.682    | 1.054     | seg3/u9   |
| DFF     | CK ^ -> Q ^   | 0.185    | 0.867    | 1.240     | seg3/u9   |
| BUF     | A ^ -> Y ^    | 0.073    | 0.940    | 1.312     | seg3/u10  |
| INV     | A ^ -> Y v    | 0.317    | 1.257    | 1.629     | seg3/u11  |
| INV     | A v -> Y ^    | 0.402    | 1.659    | 2.031     | seg3/u12  |
| BUF     | A ^ -> Y ^    | 0.403    | 2.062    | 2.434     | seg3/u13  |
| DFF     | D ^           | 0.000    | 2.062    | 2.434     | seg3/u14  |
+-------------------------------------------------------------------------+
```

The following sample timing report shows the table style generated with the `-no_frame_fix_width` parameter (`set_table_style -no_frame_fix_width`):

```
> report_timing -format {cell arc delay arrival required instance}
Path 1: MET Setup Check with Pin seg3/u14/CK
Endpoint: seg3/u14/D (^) checked with leading edge of 'CLK3'
Beginpoint: seg3/u9/Q (^) triggered by leading edge of 'CLK3'
Path Groups: {CLK3}
Other End Arrival Time 0.621
- Setup 0.186
+ Phase Shift 2.000
= Required Time 2.434
- Arrival Time 2.062
= Slack Time 0.372
Clock Rise Edge 0.000
+ Clock Network Latency (Prop) 0.682
= Beginpoint Arrival Time 0.682
-------------------------------------------------------------------
Cell      Arc           Delay    Arrival    Required    Instance
                                 Time       Time
-------------------------------------------------------------------
          CK ^                   0.682      1.054        seg3/u9
DFF       CK ^ -> Q ^   0.185    0.867      1.240        seg3/u9
BUF       A ^ -> Y ^    0.073    0.940      1.312        seg3/u10
INV       A ^ -> Y v    0.317    1.257      1.629        seg3/u11
INV       A v -> Y ^    0.402    1.659      2.031        seg3/u12
BUF       A ^ -> Y ^    0.403    2.062      2.434        seg3/u13
DFF       D ^           0.000    2.062      2.434        seg3/u14
-------------------------------------------------------------------
```

The following sample timing report shows the table style generated with the `-no_frame_fix_width` and `-nosplit` parameters (`set_table_style -no_frame_fix_width -nosplit`):

```
> report_timing -format {cell arc delay arrival required instance}
```

```
Path 1: MET Setup Check with Pin seg3/u14/CK
Endpoint: seg3/u14/D (^) checked with leading edge of 'CLK3'
Beginpoint: seg3/u9/Q (^) triggered by leading edge of 'CLK3'
Path Groups: {CLK3}
Other End Arrival Time 0.621
- Setup 0.186
+ Phase Shift 2.000
= Required Time 2.434
- Arrival Time 2.062
= Slack Time 0.372
Clock Rise Edge 0.000
+ Clock Network Latency (Prop) 0.682
= Beginpoint Arrival Time 0.682
-------------------------------------------------------------------
Cell      Arc           Delay    Arrival    Required   Instance
                                 Time       Time
-------------------------------------------------------------------
CK ^                             0.682      1.054      seg3/u9
DFF       CK ^ -> Q ^   0.185    0.867      1.240      seg3/u9
BUF       A ^ -> Y ^    0.073    0.940      1.312      seg3/u10
INV       A ^ -> Y v    0.317    1.257      1.629      seg3/u11
INV       A v -> Y ^    0.402    1.659      2.031      seg3/u12
BUF       A ^ -> Y ^    0.403    2.062      2.434      seg3/u13
DFF       D ^           0.000    2.062      2.434      seg3/u14
-------------------------------------------------------------------
```

Related Information

- report_cell_instance_timing
- report_inactive_arcs
- report_net
- report_path_exceptions
- report_ports
- report_timing

set_timing_derate

```
set_timing_derate
[-help]
[-early]
[-fall]
[-increment]
[-late]
[-min | -max | -delay_corner delayCornerName]
[-clock]
[-data]
[-power_domain string]
[-net_delay]
[-cell_delay]
[-retain_delay]
[-rise]
[-cell_check]
[-dynamic]
[-static]
[-clock_period_check]
[-pulse_width_check]
[-input_switching]
[-add]
[-mean]
[-multiply]
[-corner]
[-sigma]
[-statistical]
[-transition]
derate_value
[object_list]
```

Sets delay scaling or derating factors for early and late paths in the current design. You use this command to perform scaled on-chip variation around best-case and worst-case corners. The timing scaling factors affect the delay values generated in the timing reports. You can set scaling factors for data paths, clock paths, minimum and maximum operating conditions.

If you do not specify the `-clock` or `-data` parameter, the software applies the scaling factors to both clock and data paths. If you specify only one of the two parameters, `-data` or `-clock`, the software uses the last defined value for the other.

Similarly, if you do not specify the `-min` or `-max` parameter, the software applies the scaling factor to both minimum and maximum operating conditions. You should only use the `-min` or `-max` parameter for best-case worst-case analysis. If you specify them in on-chip variation (OCV) mode, the software will

use the `-min -early` derating factor for early path scaling, and the `-max -late` derating factor for late path adjustments.

When there are separate max and min libraries specified, only those `set_timing_derate` constraints are honored that are specified with max library. The constraints specified with min library are ignored.

In MMMC mode, for delay corners having different late and early library sets, only those `set_timing_derate` constraints are honored that are specified with late library set of the delay corner. The constraints specified with early library set are ignored.

Note: The `set_timing_derate` command can only be used on the command line. You cannot import it as part of an SDC file.

Parameters

`-help`	Prints out the command usage.
`-add`	Incrementally adds or subtracts derates. You can also apply derates on specific leaf instances.
`-cell_check`	Specifies scaling on timing checks. Use the `-cell_check` option with the `-early` option to specify scaling on hold arcs and use with the `-late` option to specify scaling on setup arcs. When you specify this parameter, derating factors are also applied to user constraints such as `set_output_delay`, `set_data_check`, and `set_clock_gating_check`. The `-cell_check` parameter by default does not apply check derate to output external delay (e.g., `set_output_delay`). To enable this feature, you can set the `timing_apply_check_derate_to_external_output_delay` attribute to `true`. When set to `true`, this attribute applies cell derate to the top boundary port.

`-cell_delay`	Specifies scaling on cell delays. If neither the `-cell_delay` nor the `-net_delay` option is specified, the specified scaling factor applies to both types of delays.
`-clock`	Applies the derating factors to clock paths only.
`-clock_period_check`	Specifies derate values for clock period checks.
`-corner`	Applies derating on cells or nets that are modeled as corner data elements. The specified derating factor is applied to both the mean and sensitivities of the delays (corner objects can have propagated sensitivities). A cell with no timing arc data in a library that has any sensitivity information with respect to any parameter defined in SPDF are considered as a corner object. All cells that do not qualify are considered to be statistical objects. A net with parasitic data that has no sensitivity information with respect to any parameter defined in SPDF is considered as a corner object. All nets that do not qualify are considered to be statistical objects.
`-data`	Applies the derating factors to data paths only.
`-delay_corner` *`delayCornerName`*	Applies the derating factor to the specified delay calculation corner or a list of delay corners. **Note**: The timing derate can be applied to corners associated with active views only.
`derate_value`	Specifies the derating value to be used.
`-dynamic`	Applies the specified derating factor to the delta portion of net delays only. You can specify `-net_delay` or `-cell_delay` parameter with the `-dynamic` parameter. You cannot specify `-dynamic` with `-mean` or `-sigma` parameters.

`-early`	Specifies that the derating factor applies to early paths. Early paths are launching clock during hold analysis, receiving clock during setup analysis and datapath during hold analysis. For definition of early and late paths, see "Definition of Early Late Paths" in *Innovus User Guide*. **Note**: To use this parameter, you must specify the `-delay_corner` parameter.
`-fall`	Applies the derating factor to falling delay and check arcs based on the transition direction at the arc's sink pin. This parameter can be used to selectively apply timing derates for the fall arcs (cell, net, or check arcs).
`-increment`	Incrementally adds the derate value to the total derate value in all the modes - OCV, AOCV, and SOCV mode.
`-input_switching`	Considers the given derate as simultaneous switching input (SSI) derate that is applicable to early paths only.
`-late`	Specifies that the derating factor applies to late paths. Late paths are receiving clock during hold analysis, launching clock during setup analysis and datapath during setup analysis. **Note**: To use this parameter, you must specify the `-delay_corner` parameter.
`-max`	Applies the derating factor to the maximum operating condition during best-case worst-case analysis. This parameter is ignored during on-chip variation analysis. **Note:** You cannot use the `-max` parameter when the software is in multi-mode multi-corner (MMMC) mode. You must use the `-delay_corner` parameter instead.

`-mean`	Controls nominal derating. You can apply separate derating the mean components of the delay.
`-min`	Applies the derating factor to the minimum operating condition during best-case worst-case analysis. This parameter is ignored during on-chip variation analysis. **Note:** You cannot use the `-min` parameter when the software is in multi-mode multi-corner (MMMC) mode. You must use the `-delay_corner` parameter instead.
`-multiply`	Incrementally multiplies derates. You can also apply derates on specific leaf instances.
`-net_delay`	Specifies scaling on net delays.
`object_list`	Specifies the list of instances, library cells, or nets to which the derating will be applied.
`-power_domain` *`string`*	Allows you to specify a specific power domain for the timing derate factors to be applied.
`-pulse_width_check`	Specifies derate values for pulse width checks. **Note:** The `-pulse_width_check` and `-clock_period_check` parameters are mutually exclusive. To apply check derating to pulse width checks, set the `timing_enable_derating_for_pulse_width_checks` attribute to `true`. By default, the attribute is set to `false`.
`-retain_delay`	Specifies scaling on retain constructs. If you specify both `-retain_delay` and `-cell_delay` parameters, the software uses the retain_delay value for retain construct scaling. If you do not specify the `-retain_delay` parameter, retain constructs are scaled in the same manner as cell delays.

`-rise`	Applies the derating factor to rising delay and check arcs based on the transition direction at the arc's sink pin. This parameter can be used to selectively apply timing derates for the rise arcs (cell, net, or check arcs).
`-sigma`	Controls derating on the variation component. You can apply separate derating to the sigma components of the delay.
`-static`	Applies the specified derating factor to the non-delta portion of net delays only. You can specify `-net_delay` or `-cell_delay` parameter with the `-static` parameter. You cannot specify `-static` with `-mean` or `-sigma` parameters.
`-statistical`	Applies derating on cells or nets that are modeled as statistical data elements. The specified derating factor is applied to both the mean and sigma of delays. If both `set_timing_derate -corner` and `-statistical` options are not specified, the derating factor is applied on variation and corner objects equally. This is equivalent to issuing to two separate `set_timing_derate` commands, one with `-statistical` option and the other with `-corner` option.
`-transition`	Specifies scaling on transitions. When specified, the software allows derate transition and stretch waveforms.

Examples

- The following example uses two `set_timing_derate` commands to set the scaling factors independently for the min and max corners. When the software is in best-case worst-case analysis mode:

  - Command (1) adds pessimism to the setup analysis by making the latching clock path 20 percent faster.
  - Command (2) adds pessimism to the hold analysis by making the latching clock path 10 percent slower.

```
(1) set_timing_derate -max -early 0.8 -late 1.0


(2) set_timing_derate -min -early 1.0 -late 1.1
```

- The following command sets the early path scaling factor of 0.85 for instance `MULT_9` cell delays:

```
set_timing_derate -early -cell_delay 0.85 [get_cells {MULT_9}]
```

- The following command sets the early path scaling factor of 0.75 for the net named `MUX2/Z`:

```
set_timing_derate -net_delay -early 0.75 [get_nets {MUX2/Z}]
```

- The following command specifies power domain `PD1` for derating instances:

```
set_timing_derate -power_domain PD1 -cell_delay 1.1
```

- In the following example, a design having max library as 'slow' and min library as 'fast', only the first two `set_timing_derate` constraints are honored:

```
set_timing_derate -late 1.2 [get_lib_cells {slow/BUFX1}] -cell_delay
set_timing_derate -early 0.8 [get_lib_cells {slow/BUFX1}] -cell_delay
set_timing_derate -late 1.1 [get_lib_cells {fast/BUFX1}] -cell_delay
set_timing_derate -early 0.9 [get_lib_cells {fast/BUFX1}] -cell_delay
```

- Consider the following library sets, in MMMC mode:

```
create_library_set -name libset1 -timing slow.lib
create_library_set -name libset2 -timing fast.lib
create_delay_corner -name corner1 -late_library_set libset1 -early_library_set
libset2
```

  From the following `set_timing_derate` constraints the first and last are honored:

```
set_timing_derate -late 1.2 -cell_delay [get_lib_cells slow/*] -delay_corner corner1
set_timing_derate -early 2.3 -cell_delay [get_lib_cells slow/*] -delay_corner corner1
set_timing_derate -late 3.4 -cell_delay [get_lib_cells fast/*] -delay_corner corner1
set_timing_derate -early 4.5 -cell_delay [get_lib_cells fast/*] -delay_corner corner1
```

- The following command can be used when the software is in on-chip variation analysis mode to add pessimism to both the setup and hold analyses by making the early paths 20 percent faster:

```
set_analysis_mode -analysisType onChipVariation
...
```

```
set_timing_derate -early 0.8 -late 1.0
```

This command adds pessimism to the setup analysis by making the latching clock path 20 percent faster.

- The following example uses two `set_timing_derate` commands to set the scaling factors independently for the min and max corners. When the software is in best-case worst-case analysis mode:
  - The following command sets the early path scaling factor of 0.85 for instance MULT_9 cell delays:
    ```
    set_timing_derate -early -cell_delay 0.85 [get_cells {MULT_9}]
    ```
  - The following command sets the early path scaling factor of 0.75 for the net named MUX2/Z:
    ```
    set_timing_derate -net_delay -early 0.75 [get_nets {MUX2/Z}]
    ```
- The following command specifies power domain PD1 for derating instances:
  ```
  set_timing_derate -power_domain PD1 -cell_delay 1.1
  ```
- The following command shows library object level support for clock (and pulse width) check options:
  ```
  set_timing_derate -delay_corner [all_delay_corners] -clock_period_check 1.05
  [object_list]
  ```

Related Commands

- `report_timing_derate`
- `reset_timing_derate`

set_wire_load_mode

```
set_wire_load_mode
[-help]
{top | enclosed | segmented}
```

Controls how the software searches through the design hierarchy to find the appropriate wire load model for a net, or a hierarchical part of a net.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `set_wire_load_mode` parameter. For a detailed description of the command and all of its parameters, use the man command: `man set_wire_load_mode`.
*`{top	enclosed	segmented}`*	Specifies the wire load model. You can specify one of the following options. • `enclosed` • `segmented` • `top`

	`enclosed`: Selects the wire load model using an area lookup table for the module that is at the lowest level in the design hierarchy and contains the entire net. When you set the mode to `enclosed`, the software uses the area of the smallest block enclosing the net to select the wire load model. In the `enclosed` mode, the software uses the following order of precedence to search the wire load model in: • The inner-most block enclosing the net on which you set the wire load selection group using the `set_wire_load_selection_group` command, if the `auto_wire_load_selection` global variable is enabled. Or: The inner-most block enclosing the net on which you set the wire load model using the `set_wire_load_model` command. • Each higher level hierarchical block for a `set_wire_load_selection_group` assertion, all the way up to the top-level block, if the `auto_wire_load_selection` global variable is enabled. Or: Each higher level hierarchical block for a `set_wire_load_model` assertion, all the way up to the top-level block. • The `default_wire_load_selection` field in the first timing library. Or, The `default_wire_load` field in the first timing library. **Note:** If even the default values are not present in the timing library, you will not be able to use the wireload model functionality.

	`segmented`: Selects the wire load model in a way similar to the `enclosed` mode except that each hierarchical segment of the net is treated separately. In the `segmented` mode: • A segment is defined by pins (including hierarchical pins) connected to a particular net. • When counting the drivers and receivers, only the pins connected to the segment are considered. This information along with the wire load models determine the parasitic information for the segment. • The parasitics of each segment sum up to form the parasitics for the entire net. **Note:** Each segment may have different wire load models.
	`top`: Selects the wire load model in a way similar to `enclosed` and `segmented` except that all nets use the wire load selection or wire load models defined by the top-level (largest) block enclosing the net. **Note:** If the first library read in does not define the default wire load mode, the software uses `top` as the default mode.

Examples

- The following command sets the wire load mode to `enclosed`:

```
set_wire_load_mode enclosed
```

Related Commands

- `reset_wire_load_mode`

set_wire_load_model

```
set_wire_load_model
[-help]
[obj_list]
[-library library_name]
[-max]
[-min]
-name wireload_model
```

Specifies the wire load model to be used from the technology library, and sets the wire load model on the design or on a hierarchical object (instance). If the list of instances is not specified, the wire load model is set on the current instance. Otherwise, the wire load model is set on all the specified instances or ports.

Use wire load models for estimating delays before the actual wire loads are back annotated. A technology library contains different wire load models, previously computed based on the analysis of several designs differing in area.

Parameters

`-help`	Prints out the command usage.
`-library` *library_name*	Specifies the technology library to use when searching for the name of the wire load model. You can specify a library name, or a collection of libraries. *Default:* All technology libraries are searched.
obj_list	Specifies the list of objects - designs, ports, or hierarchical instances.
`-max`	Sets the wire load model only for the maximum PVT (process, voltage, temperature) corner. *Default*: The specified wire load model applies to all three PVT corners if neither the `-min` nor the `-max` parameter is specified.
`-min`	Sets the wire load model only for the minimum PVT (process, voltage, temperature) corner. *Default*: The specified wire load model applies to all three PVT corners if neither the `-min` nor the `-max` parameter is specified.
`-name` *wireload_model*	Specifies the name of the wire load model in the library.

Examples

- The following command sets the `COM3_LV_WLM_0.0` wire load model from the `COM3_LV_WIRE` library on the current module for all PVT corners:

  ```
  set_wire_load_model -library COM3_LV_WIRE -name COM3_LV_WLM_0.0
  ```

- The following command sets the `B1X1` wire load from the `lca500kv` library on the `clk1` port for the minimum PVT corner:

  ```
  set_wire_load_model -library lca500kv -min -name B1X1 clk1
  ```

- The following command sets the `3M_TO_4500K`wire load from the `c13tsmc_slow2s108v125c` library on the current module for the minimum PVT corner:

  ```
  set_wire_load_model -library c13tsmc_slow2s108v125c -min -name 3M_TO_4500K
  ```

Related Commands

- `report_wire_load`
- `reset_wire_load_model`

set_wire_load_selection_group

```
set_wire_load_selection_group
[-help]
group_name
[hinst_list]
[-library library_name]
[-max]
[-min]
```

Overrides the wire load selection table in the specified library.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `set_wire_load_selection_group` parameter. For a detailed description of the command and all of its parameters, use the man command: `man set_wire_load_selection_group`
group_name	Specifies the name of the wire load selection group.
`-library` *library_name*	Specifies the name of the technology library to search for the wire load selection table. When this parameter is not specified, the software searches all the libraries.
hinst_list	Specifies the list of instances.
`-max`	Controls the choice of a wire load selection table only for the PVT max corner. *Default*: Selects the table for all PVT corners if neither the `-min` nor the `-max` parameter is specified.

`-min`	Controls the choice of a wire load selection table only for the minimum PVT corner. *Default*: Selects the table for all PVT corners if neither the `-min` nor the `-max` parameter is specified.

Examples

- The following command selects `TLM` as the selection table used by the library:
  ```
  set_wire_load_selection_group TLM
  ```
- The following command selects `ASPEC1` as the selection table used by the `sm333s` library; only the minimum PVT corner is affected:
  ```
  set_wire_load_selection_group -min -library sm333s ASPEC1
  ```

Related Commands

- `reset_wire_load_selection_group`

time_design

```
time_design
[-help]
[-drv_report]
[-expand_reg2reg]
[-expanded_views]
[-hold]
[-ideal_clock]
[-num_paths num_paths]
[-path_report]
[-pre_place]
[-pre_cts]
[-post_cts]
[-post_route]
[-prototype]
[-report_dir string]
[-report_only]
[-report_prefix <string>]
[-slack_report]
[-timing_debug_report]
[-use_transition_files]
```

Runs early global route, extraction, and timing analysis, and generates detailed timing reports. The generated timing reports are saved in `./timingReports` directory or the directory that you specify using the `-report_dir` parameter.

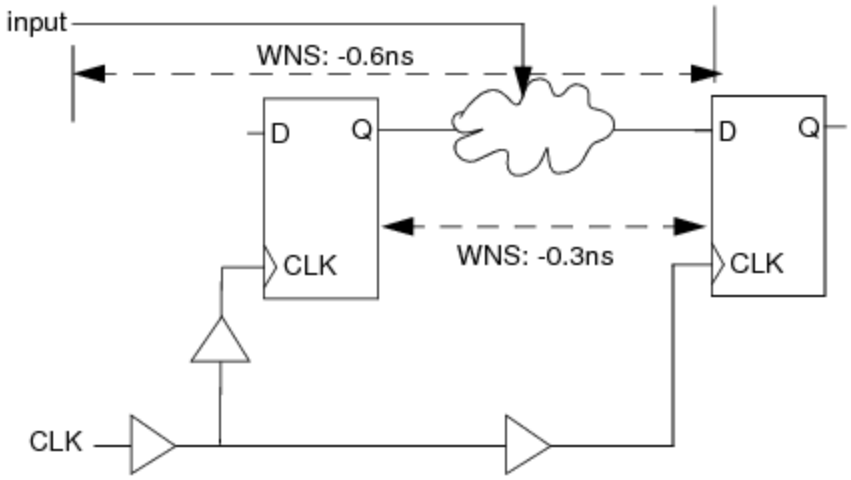
The following timing reports are generated:

- Setup and hold time for the following path groups:
    - Register to register (*TopCellName_DesignStage*`_reg2reg.tarpt`)
    - Input to register (*TopCellName_DesignStage*`_in2reg.tarpt`. This report also includes the clock source paths (if applicable to the design). The clock source paths are included even if the clock source is an internal pin.
    - Register to output (*TopCellName_DesignStage*`_reg2out.tarpt`)
    - Input to output (*TopCellName_DesignStage*`_in2out.tarpt`)
    - All paths ending in clock gate (*TopCellName_DesignStage*`_clkgate.tarpt`)
      For hold analysis, the report names are *TopCellName_DesignStage_PathGroup_hold*`.tarpt`
- Setup and hold time without using path groups:

    - Setup analysis (*TopCellName_DesignStage*_all.tarpt)
    - Hold analysis (*TopCellName_DesignStage*_all_hold.tarpt)
- Density
  Note: The density value reported by the time_design command might differ from the effective utilization value reported by the check_place command because time_design does not consider padding when calculating density, but check_place does consider it when calculating effective utilization.
- Maximum transition time violation (*TopCellName_DesignStage*.tran): The maximum transition value reported is the maximum value from set_max_transition command from SDC or the max_transition value specified in the library.
- Maximum capacitance violation (*TopCellName_DesignStage*.cap): The maximum capacitance value reported is the maximum value from the set_max_capacitance command from SDC or the max_capacitance value specified in the library.
- Maximum fanout violation (*TopCellName_DesignStage*.fanout)
- Glitch violations in the *outDir*/*prefix_hold*/setup directory.
- Summary (*TopCellName_DesignStage*.summary and *TopCellName_DesignStage*_hold.summary)
  Note: The ID number displayed in the Summary report refers to long path group names.

The overall total negative slack (TNS) and number of violating paths of a design might not be equal to the total of the TNS and violating paths of the individual path groups. This is because the TNS and number of violating paths are based on end-point of the path and are not path based.

For example, the following figure has a register with two paths, one from a primary input with a slack of -0.6ns and other from another register with a slack of -0.3ns.



In that case the overall TNS will be `-0.6ns` with `1` violating path (end point-based). But the individual reg2reg TNS will be `-0.3ns` with `1` violating path and in2reg with a TNS of `-0.6ns` with `1` violating path. Therefore, the sum total of individual path group TNS is not the same as overall TNS.

Similarly, when you run the `time_design` command while performing multi-mode multi-corner analysis, overall TNS is not the sum of TNS of each view considered separately. The command sums the worst-case slack at each endpoint across all views. Therefore each endpoint contributes only in overall worst-slack in the calculation of TNS.

Parameters

`-help`	Prints out the command usage.
`-drv_report`	Generate design rule violation (DRV) reports only. *Default:* Generates DRV, slack, and path reports.
`-expanded_views`	Generates detailed view-specific timing reports when the software is in multi-mode multi-corner analysis mode. For multi-mode multi-corner analysis, the software creates a separate directory for each view. For example, if your design has two analysis views, `view1` and `view2`, the output reports are generated in the `./timingReports/view1` and `./timingReports/view2` directories.
`-expand_reg2reg`	Specifies that the register to register path groups are divided into the following detailed path groups for generating reports: • Flop to flop • Macro to flop • Flop to macro • Macro to macro
`-hold`	Reports hold violations only. Before ending the `time_design` run, the software reverts back to the initial analysis mode that was set using the timing Category Attributes. *Default:* Reports only setup violations

`-ideal_clock`	Performs ideal clock analysis irrespective of the design stage. *Default:* The software performs propagated clock analysis with the `-post_cts` and `-post_route` parameters. The software performs ideal clock analysis with the `-pre_place` and `-pre_cts` parameters. **Note:** This parameter is not honored when used with `-report_only`.
`-num_paths` *num_paths*	Specifies the number of paths to report in the detailed path violations. *Default:* `50`
`-report_dir` *string*	Specifies the name of the output directory that contains all the generated reports. *Default:* `timingReports`
`-path_report`	Generate path reports only. *Default:* Generates DRV, slack, and path reports.
`-pre_cts`	Generates timing reports before the clock tree is built. By default, in this mode the software runs early global route, native default extraction, and timing analysis for generating timing reports. The `-pre_cts` parameter sets `timing_analysis_clock_source_paths` attribute to `false` and `timing_analysis_clock_propagation_mode` attribute to `forced_ideal`.
`-pre_place`	Generates timing reports before the design is placed. In this mode, the software ignores the net load while building the timing graph and runs timing analysis for generating timing reports. The DRV report files are not generated in this mode, and the DRV numbers are not reported in the summary for this parameter. The `-pre_place` parameter sets `timing_analysis_clock_source_paths` attribute to `false` and `timing_analysis_clock_propagation_mode` attribute to `forced_ideal`. **Note:** The `-report_only` parameter is not honored when used with this parameter.

`-prototype`	Supports flex models. Notes: • The `-prototype` parameter supports `set_proto_mode -timing_net_delay_model option`. • The `-prototype` parameter honors the `-post_cts` option.
`-post_cts`	Generates timing reports for a design whose clock tree has been created. By default, in this mode the software runs early global route, native default extraction, and timing analysis for generating timing reports. The `-post_cts` parameter sets `timing_analysis_clock_source_paths` attribute to `true` and `timing_analysis_clock_propagation_mode` attribute to `sdc_control`.
`-post_route`	Generates timing reports for a design whose routing is complete. By default, in this mode the software runs native detailed extraction, and timing analysis for generating timing reports. The `-post_route` parameter sets `timing_analysis_clock_source_paths` attribute to `true` and `timing_analysis_clock_propagation_mode` attribute to `sdc_control`.
`-report_prefix` *`string`*	Specifies a prefix for the timing report file names. *Default:* *`DesignName_DesignStage`*
`-report_only`	Specifies the use of existing extraction and timing analysis data to generate timing reports. When you use this parameter, the software does not run extraction; instead it uses data that is already in memory. • `-report_only` is not honored when used with `-pre_place`. • `-ideal_clock` is not honored when used with `-report_only`. **Note:** The `-pre_cts`, `-post_cts`, and `-post_route` parameters are not compatible with `-report_only`.
`-slack_report`	Generate slack reports only. *Default:* Generates DRV and path reports.
`-timing_debug_report`	Generates a compressed (`.gz`) detailed timing report in ASCII format. This report is used for debugging timing results using the timing debug feature in Stylus.

`-use_transition_files`	Reports the maximum transition violations when using external transition files specified by the `read_transition_file` command.

Command Order

Use this command after loading the design.

Examples

- The following command runs early global route, native default extraction, and timing analysis to generate timing reports before clock tree synthesis:

  ```
  time_design -pre_cts
  ```

- The following example shows a summary report generated by `time_design`:

  ```
  # Unit: lps
  #     <late_rise>     <late_fall>    <early_rise>    <early_fall> <hpin_name>
  # ------------------------------------------------------------------------------
        -4176.76        -4049.26       -82696.70       -82823.75    1_pin_name
        -7450.33        -6944.72       -82284.92       -82297.52    2_pin_name
        -4128.03        -3988.92       -82745.17       -82883.96    3_pin_name
        -5683.80        -5937.10       -86789.41       -86535.58    4_pin_name
  ```

- The following command ignores the net load while building the timing graph and runs timing analysis for generating timing reports before the design is placed:

  ```
  time_design -pre_place
  ```

- The following commands run native detailed extraction, and timing analysis to generate timing reports for both setup and hold violations after the design has been routed:

  ```
  time_design -post_route
  time_design -hold -report_only
  ```

- The following command generates timing reports in multi-mode multi-corner timing analysis view, where test1 is the specified analysis view:

  ```
  set_analysis_view -setup test1 -hold test1
  time_design -post_route -report_dir sample
  ```

  The following summary report is generated:

  ```
  time_design Summary
  ```

```
Setup mode     all     reg2reg    in2reg   reg2out   in2out   clkgate

WNS (ns):     -10.821  -7.001     -7.264    -4.978    -10.821   N/A

TNS (ns):      -6898.5 -5609.5    -1528.6    -55.664   -150.330 N/A

Violating Paths:4836    2756      2891       16        16       N/A
All Paths:      6866    6698      5654       84       16        N/A

test1          -10.821  -7.001    -7.264    -4.978     -10.821  N/A

               -6898.5  -5609.5   -1528.6    -55.664   -150.330 N/A

               4836      2756      2891     16          16      N/A

               6866      6698      5654      84         16      N/A

Density: 69.829%

Real DRV (fanout, cap, tran): (2, 524, 104)

Total DRV (fanout, cap, tran): (10, 524, 104)
```

Related Topics

- check_place
- read_transition_file
- set_analysis_view
- set_table_style
- set_max_capacitance
- set_max_transition
- Accelerating the Design Process By Using Multiple-CPU Processing chapter of the *Innovus Stylus Common UI User Guide*

time_design_signoff

```
time_design_signoff
[-help]
[-no_eco_db]
[-no_expanded_views]
[-report_dir string]
[-report_full_clock_path]
[-report_only]
[-report_prefix string]
```

Runs signoff timing analysis using extraction (Quantus) and Tempus in batch mode, and generates timing reports and ECO DB for each view. The command prints out a timing summary table per view and all the views combined.

Parameters

`-help`	Prints out the command usage.
`-no_eco_db`	Disables generation of ECO timing DB. *Data_type*: bool, optional
`-no_expanded_views`	Disables printing of expanded timing per view. *Data_type*: bool, optional
`-report_dir string`	Specifies the directory to save reports. *Data_type*: string, optional
`-report_full_clock_path`	When this parameter is specified, the detailed timing report that is generated contains the full clock path. *Data_type*: bool, optional
`-report_only`	Skips parasitic extraction. *Data_type*: bool, optional
`-report_prefix string`	Specifies a prefix for timing report file names. *Data_type*: string, optional

Related Information

- `report_timing`
- `time_design`
- The following section in Optimizing Timing chapter of *Innovus Stylus Common UI User Guide*
  - Running MMMC SignOff ECO within Innovus
  - Optimization in Path-Based Analysis (PBA) Mode
  - Path Group Support

timing Category Attributes

- timing_all_registers_include_icg_cells
- timing_allow_input_delay_on_clock_source
- timing_analysis_aocv
- timing_analysis_async_checks
- timing_analysis_case_analysis
- timing_analysis_check_type
- timing_analysis_clock_gating
- timing_analysis_clock_net_marking_mode
- timing_analysis_clock_propagation_mode
- timing_analysis_clock_source_paths
- timing_analysis_cppr
- timing_analysis_engine
- timing_analysis_honor_active_logic_view
- timing_analysis_self_loops_paths_no_skew
- timing_analysis_precision_ps
- timing_analysis_socv
- timing_analysis_type
- timing_aocv_analysis_mode
- timing_aocv_chip_size
- timing_aocv_core_size
- timing_aocv_derate_mode
- timing_aocv_stage_count_update_on_timing_reset
- timing_apply_check_derate_to_external_output_delay
- timing_apply_default_primary_input_assertion
- timing_apply_exceptions_to_data_check_related_pin
- timing_apply_setup_hold_exceptions_to_data_check_related_pin

- timing_cap_unit
- timing_case_analysis_propagation
- timing_case_analysis_for_sequential_propagation
- timing_case_analysis_for_icg_propagation
- timing_check_timing_report_all_checks
- timing_check_timing_signal_level_high_to_low_threshold
- timing_check_timing_signal_level_low_to_high_threshold
- timing_clock_phase_propagation
- timing_clock_source_paths_unconstrained_mark_clock_used_as_data
- timing_clock_source_use_driving_cell
- timing_clock_uncertainty_from_to_precedence
- timing_collection_all_fanin_fanout_traversal_mode
- timing_collection_result_display_limit
- timing_collection_variable_assignment_compatibility
- timing_constraint_disable_min_max_input_delay_worst_casing
- timing_constraint_enable_detailed_report_invalid_begin_end_points
- timing_constraint_enable_drv_limit_override
- timing_constraint_enable_logging
- timing_constraint_enable_report_invalid_begin_end_points
- timing_constraint_enable_search_path
- timing_constraint_enable_separate_multicycle_data_checks
- timing_constraint_load_minimal_set_for_automated_eco
- timing_constraint_path_delay_exclude_check_delay_from_ignore_clock_latency
- timing_constraint_path_delay_exclude_io_delay_from_ignore_clock_latency
- timing_constraint_path_delay_exclude_unconstrained_endpoints
- timing_constraint_path_delay_include_clock_pin_endpoints
- timing_constraint_warn_for_timing_derate_exceeding_max_limit
- timing_continue_on_error

- timing_constraint_update_io_latency_averaging_mode
- timing_context_clock_mapping_percentage_tolerance
- timing_context_clock_mapping_check_waveform
- timing_cppr_enable_mismatch_transition_mode
- timing_cppr_opposite_edge_mean_scale_factor
- timing_cppr_opposite_edge_sigma_scale_factor
- timing_cppr_opposite_edge_sigma_scale_factor_net
- timing_cppr_opposite_edge_sigma_scale_factor_cell
- timing_cppr_propagate_thru_latches
- timing_cppr_remove_clock_to_data_pessimism
- timing_cppr_self_loop_mode
- timing_cppr_skip_clock_reconvergence
- timing_cppr_skip_clock_reconvergence_for_unmatched_clocks
- timing_cppr_threshold_ps
- timing_cppr_transition_sense
- timing_create_clock_default_propagated
- timing_default_opcond_per_lib
- timing_defer_mmmc_obj_updates
- timing_derate_aocv_dynamic_delays
- timing_derate_aocv_reference_point
- timing_derate_dynamic_compatibility
- timing_derate_ocv_reference_point
- timing_derate_negative_delay_backward_compatibility
- timing_derate_spatial_distance_unit
- timing_derate_incremental_adjust_additive_mode
- timing_disable_bus_contention_check
- timing_disable_clock_period_checks
- timing_disable_constant_propagation_for_sequential_cells

- timing_disable_drv_report_on_constant_nets
- timing_disable_floating_bus_check
- timing_disable_genclk_combinational_blocking
- timing_disable_inferred_clock_gating_checks
- timing_disable_inout_output_side_timing_checks
- timing_disable_internal_inout_cell_paths
- timing_disable_internal_inout_net_arcs
- timing_disable_lib_pulse_width_checks
- timing_disable_library_data_to_data_checks
- timing_disable_library_tieoffs
- timing_disable_netlist_constants
- timing_disable_nochange_checks
- timing_disable_non_sequential_checks
- timing_disable_output_as_clock_port
- timing_disable_pulse_width_same_edge_si_cppr_mode
- timing_disable_report_header_info
- timing_disable_retime_clock_path_slew_propagation
- timing_disable_sdf_retain_arc_merging
- timing_disable_skew_checks
- timing_disable_test_signal_arc
- timing_disable_timing_model_latch_inferencing
- timing_disable_tristate_disable_arcs
- timing_disable_user_data_to_data_checks
- timing_driving_cell_override_library
- timing_enable_aocv_slack_based
- timing_enable_case_analysis_conflict_warning
- timing_enable_clock_to_clock_clock_gating_check
- timing_enable_clock_phase_based_rise_fall_derating

- timing_enable_data_through_clock_gating
- timing_enable_derating_for_pulse_width_checks
- timing_enable_early_late_data_slews_for_setuphold_mode_checks
- timing_enable_genclk_divide_by_inherit_parent_duty_cycle
- timing_enable_genclk_source_path_register_limit
- timing_enable_generated_clock_edge_based_source_latency
- timing_enable_get_obj_escaped_name_backward_compatible
- timing_enable_get_objects_regexp_compatibility
- timing_enable_hierarchical_get_nets_support
- timing_enable_latch_thru_mode
- timing_enable_latency_through_clock_gating
- timing_enable_mmmc_loop_breaking
- timing_enable_multi_drive_net_reduction_with_assertions
- timing_enable_multi_frequency_latch_analysis
- timing_enable_multicycle_data_check_compatibility
- timing_enable_path_delay_to_unconstrained_endpoints_compatibility
- timing_enable_pessimistic_cppr_for_reconvergent_clock_paths
- timing_enable_power_ground_constants
- timing_enable_preset_clear_arcs
- timing_enable_pulse_latch
- timing_enable_si_cppr
- timing_enable_simultaneous_setup_hold_mode
- timing_enable_timing_window_pessimism_removal
- timing_enable_tristate_clock_gating
- timing_enable_uncertainty_for_clock_checks
- timing_enable_uncertainty_for_pulse_width_checks
- timing_extract_model_disable_cycle_adjustment
- timing_extract_model_aocv_mode

- timing_extract_model_case_analysis_in_library
- timing_extract_model_check_arcs_as_lvf
- timing_extract_model_consider_design_level_drv
- timing_extract_model_exhaustive_validation_dir
- timing_extract_model_exhaustive_validation_mode
- timing_extract_model_gating_as_nochange_arc
- timing_extract_model_ideal_clock_latency_arc
- timing_extract_model_include_applied_load_in_characterization_range
- timing_extract_model_include_applied_slew_in_characterization_range
- timing_extract_model_non_borrowing_latch_path_as_setup
- timing_extract_model_slew_propagation_mode
- timing_extract_model_write_clock_checks_as_arc
- timing_extract_model_write_clock_checks_as_scalar_tables
- timing_extract_model_write_lvf
- timing_extract_model_write_min_max_clock_tree_path
- timing_generated_clocks_inherit_ideal_latency
- timing_generated_clocks_allow_nested_assertions
- timing_generate_normalized_driver_waveform
- timing_get_of_objects_hier_compatibility
- timing_hier_obj_name_compatibility
- timing_ignore_lumped_rc_assertions
- timing_io_use_clock_network_latency
- timing_library_build_async_deassert_arc
- timing_library_convert_async_setuphold_to_recrem
- timing_library_generated_clock_use_group_name
- timing_library_hold_constraint_corner_sigma_multiplier
- timing_library_hold_sigma_multiplier
- timing_library_infer_async_pins_from_timing_arcs

- timing_library_infer_cap_range_from_ccs_receiver_model
- timing_library_infer_cap_range_from_ecsm_receiver_model
- timing_library_interpolate_drv_values
- timing_library_infer_socv_from_aocv
- timing_library_read_ccs_noise_data
- timing_library_read_without_ecsm
- timing_library_read_without_power
- timing_library_read_without_sensitivity
- timing_library_scale_aocv_to_socv_to_n_sigma
- timing_library_setup_constraint_corner_sigma_multiplier
- timing_library_setup_sigma_multiplier
- timing_library_term_voltage_from_lib_pin
- timing_library_use_two_piece_receiver_cap
- timing_library_zero_negative_timing_check_arcs
- timing_multi_frequency_clock_rounding_factor
- timing_normalized_driver_waveform_clip_linear_part
- timing_normalized_driver_waveform_weight_factor
- timing_nsigma_multiplier
- timing_null_collection_return_compatibility
- timing_path_based_enable_bounding_box_for_io_paths
- timing_path_based_exhaustive_pba_bounded_mode
- timing_path_based_enable_closest_common_pin_driver_for_bounding_box
- timing_path_based_enable_exhaustive_depth_bounded_by_gba
- timing_path_based_enable_report_launch_clock_path
- timing_path_based_enable_verbose_mode
- timing_path_based_exhaustive_enable_design_coverage
- timing_path_based_exhaustive_max_paths_limit
- timing_pba_exhaustive_path_nworst_limit

- timing_pll_clock_use_driving_cell
- timing_prefix_module_name_with_library_generated_clock
- timing_propagate_latch_data_uncertainty
- timing_property_arrival_clocks_consider_clock_source_data_phase
- timing_property_arrival_clocks_consider_data_phase
- timing_property_arrival_clocks_consider_latency_phase
- timing_property_arrival_window_enable_tcl_dict_format
- timing_property_return_null_collection_with_quiet
- timing_rail_swing_checks_high_voltage_threshold
- timing_rail_swing_checks_low_voltage_threshold
- timing_recompute_sdf_in_setuphold_mode
- timing_reduce_multi_drive_net_arcs
- timing_reduce_multi_drive_net_arcs_threshold
- timing_report_analysis_summary_csv_extended_new_format
- timing_report_check_timing_unconstrained_endpoints_due_to_constants
- timing_report_clock_pin_as_begin_point
- timing_report_constraint_enable_extended_drv_format
- timing_report_constraint_rise_fall_clock_period_check
- timing_report_disable_max_paths_per_group
- timing_report_drv_enable_clock_source_as_clock
- timing_report_drv_enable_frequency_per_view
- timing_report_drv_enable_slew_threshold_scaling
- timing_report_enable_cppr_point
- timing_report_enable_flag_field_symbols
- timing_report_enable_lead_trail_to_rise_fall_map
- timing_report_enable_markers
- timing_report_enable_max_capacitance_drv_for_constant_nets
- timing_report_enable_max_path_limit_warning

- timing_report_enable_report_clock_timing_across_clock_pin
- timing_report_enable_si_debug
- timing_report_enable_unique_pins_multiple_capture_clock_paths
- timing_report_enable_verbose_ssta_mode
- timing_report_fields
- timing_report_generated_clock_info
- timing_report_group_based_mode
- timing_report_max_transition_check_using_nsigma_slew
- timing_report_property_fastest_clock_consider_data_phase
- timing_report_pulse_width_matching_launch_capture_paths
- timing_report_redirect_message_types
- timing_report_retime_formatting_mode
- timing_report_skip_constraint_loop_check
- timing_report_socv_summary_mean_sigma
- timing_report_timing_header_detail_info
- timing_report_unconstrained_paths
- timing_report_unconstrained_path_early_late_header
- timing_report_use_receiver_model_capacitance
- timing_report_use_worst_parallel_cell_arc
- timing_resolve_driver_conflicts
- timing_scaling_for_negative_checks
- timing_scaling_for_negative_delays
- timing_sdf_adjust_negative_setuphold
- timing_sdf_enable_setuphold_scond_ccond
- timing_self_loop_paths_no_skew_max_depth
- timing_self_loop_paths_no_skew_max_slack
- timing_set_clock_source_to_output_as_data
- timing_socv_preserve_variation_with_annotations

- timing_socv_rc_variation_mode
- timing_socv_statistical_min_max_mode
- timing_socv_view_based_nsigma_multiplier_mode
- timing_spatial_derate_chip_size
- timing_spatial_derate_distance_mode
- timing_enable_spatial_derate_mode
- timing_suppress_escape_characters
- timing_suppress_ilm_constraint_mismatches
- timing_time_unit
- timing_use_clock_pin_attribute_for_clock_net_marking
- timing_use_incremental_si_transition
- timing_use_latch_early_launch_edge
- timing_use_latch_time_borrow
- timing_use_verilog_for_model_netlist
- timing_waveform_aware_pulse_width_checks_high_voltage_level
- timing_waveform_aware_pulse_width_checks_low_voltage_level
- timing_write_sdf_no_escape_backslash

To set the value of any root attribute, use the `set_db` command:

```
set_db attribute_name value
```

To get the current value of any root attribute, use the `get_db` command:

```
get_db attribute_name
```

To get the current value of all attributes in the `check_drc` category, use the following command:

```
get_db -category check_drc
```

timing_all_registers_include_icg_cells

`timing_all_registers_include_icg_cells {true | false}`

Default: `true`

Includes integrated clock gating (ICG) cell information from the `all_registers` report output.

When set to `false`, the global does not allow the `all_registers` report output to display ICG cell details.

timing_allow_input_delay_on_clock_source

```
timing_allow_input_delay_on_clock_source   {true | false}
```

Default: `false`

When set to `true`, this attribute allows the software to apply input delay constraints on a pin where a clock was previously asserted.

timing_analysis_aocv

```
timing_analysis_aocv {true | false}
```

Default: `false`

Sets the analysis mode to advanced on-chip variation (AOCV) analysis.

timing_analysis_async_checks

```
timing_analysis_async_checks {async |no_async | async_only}
```

Default: `async`

Reports timing violations for asynchronous timing checks, including recovery and removal, and other checks on pins defined as preset or clear in the Liberty file.

The following options are supported:

- `async`: Reports timing checks on pins that are defined as preset or clear in the Liberty file.
- `no_async`: Reports timing violations while ignoring the two time arc attributes.
- `async_only`: Reports only recovery and removal checks.

The following setting corresponds to `set timing_disable_recovery_removal_checks true` in Innovus legacy.

```
set_db timing_analysis_async_checks no_async
```

The following setting corresponds to `set timing_disable_recovery_removal_checks false` in Innovus legacy.

```
set_db timing_analysis_async_checks async
```

timing_analysis_case_analysis

`timing_analysis_case_analysis {true | false}`

Default: `true`

Determines whether constants supplied from the constraint file and netlist should be applied to the analysis.

When you specify `false`, the software blocks the application of constants from the timing constraints file. However, constants that are introduced through the netlist and timing library are still applied and propagated.

timing_analysis_check_type

`timing_analysis_check_type {setup | hold}`

Default: `setup`

Checks the design for setup or hold violations in the current analysis.

Note: When simultaneous setup/hold mode analysis is on, this parameter does not have any impact. You can use the `report_timing -late/-early` parameter to report setup/hold timing reports.

timing_analysis_clock_gating

`timing_analysis_clock_gating {true | false}`

Default: `true`

Reports the gating checks.

timing_analysis_clock_net_marking_mode

`timing_analysis_clock_net_marking_mode {before_constant_propagation | after_constant_propagation}`

Default: `before_constant_propagation`

Marks clock nets before or after propagating the constants.

timing_analysis_clock_propagation_mode

timing_analysis_clock_propagation_mode {`sdc_control` | `forced_ideal`}

Default: `sdc_control`

Specifies how to determine whether clock end points are interpreted as having either ideal or propagated clock timing.

- `sdc_control`: Reports the timing violations considering clock tree and enables propagating clocks to report clock latency. You must set the `set_propagated_clock` constraint to propagate the clocks.
  Note: By default, CTS will instruct the timer to make all of the clock roots propagated. In this case, you do not need to set the `set_propagated_clock` constraint.
- `forced_ideal`: Does not report the timing violations considering the clock tree.

timing_analysis_clock_source_paths

`timing_analysis_clock_source_paths {true | false}`

Default: `true`

Reports combinational paths from the clock source to the timing checks on data pins.

When set to `false`, the clock portion of the path is considered as ideal (due to the SDC constraints or the `timing_analysis_clock_propagation_mode forced_ideal` attribute setting), the delay calculation software treats the clock nets as ideal nets when calculating the delays and slews along the data path.

For all pre-clock tree synthesis (CTS), `optDesign`, and `timeDesign` steps, the software uses the `false` setting in combination with the `timing_analysis_clock_propagation_mode forced_ideal` setting to calculate the delays and slews along the data path.

For all post-CTS flow steps, the software uses the default setting in combination with `timing_analysis_clock_propagation_mode sdc_control` to calculate the delays and slews along the data path.

timing_analysis_cppr

`timing_analysis_cppr {both | none | setup | hold}`

Default: `none`

Removes pessimism from clock paths that have a portion of the clock network in common between the clock source and clock destination paths.

The pessimism is introduced when the timing analysis tools assume that the common path has different delay values for two different paths in case of on-chip variation. The pessimism can also be because of clock reconvergence in the clock network. The pessimism effects all analysis modes

including single, best case-worst case (bcwc), and on-chip variation.

- `none`: Disables removal of clock reconvergence pessimism.
- `both`: Enables removal of clock reconvergence pessimism for both setup and hold modes.
- `setup`: Enables removal of clock reconvergence pessimism in setup mode only.
- `hold`: Enables removal of clock reconvergence pessimism in hold mode only.

timing_analysis_engine

```
timing_analysis_engine {statistical | static}
```

Default: `static`

Allows you to specify the analysis engine for performing timing analysis. You can specify the following options:

- `statistical`: Sets analysis mode to SSTA (statistical static timing analysis).
- `static`: Performs static timing analysis.

timing_analysis_honor_active_logic_view

```
timing_analysis_honor_active_logic_view {true | false}
```

Default: `false`

Sets the timing analysis mode to honor Active Logic View (previously referred to as CDTV and Virtual Partition). Instead of analyzing the whole chip, the software analyzes only the nets related to interface paths between the partitions.

Note: This attribute applies to pre-CTS mode only.

timing_analysis_self_loops_paths_no_skew

```
timing_analysis_self_loops_paths_no_skew {true | false}
```

Default: `false`

Checks for self loop paths without skew.

When set to `true`, the software eliminates clock skew due to clock uncertainty for a path starting and ending at the same register. If the clock skew is not eliminated, the timing for such paths is pessimistic.

Note: Use this attribute before building the timing graph.

timing_analysis_precision_ps

`timing_analysis_precision_ps {0.1 | 0.01}`

Default: `0.01` (in ps)

Allows you to set the delay precision value for the `write_sdf` command.

The `write_sdf` command shows a double digit value after the decimal point by default. You can use this attribute for more accuracy.

timing_analysis_socv

`timing_analysis_socv {true | false}`

Default: `false`

Enables SOCV Analysis.

timing_analysis_type

`timing_analysis_type {single | best_case_worst_case | ocv}`

Default: `ocv`

Sets the timing analysis type to single, best case worst case or on-chip variation.

In this mode, the software calculates the delay for one path based on maximum operating condition while calculating the delay for another path based on minimum operating condition for setup or hold checks.

- `single`: Scales the delay values based on one operating condition. For more information, see "Single Timing Analysis Mode" in the *Innovus User Guide*.
- `best_case_worst_case`: Checks the design for two extreme operating conditions. The software uses the maximum delays for all paths during setup checks and minimum delays for all paths during hold checks. For more information, see "Best-Case Worst-Case (BC-WC) Timing Analysis Mode," in the *Innovus User Guide*.
- `ocv`: Calculates the delay for one path based on maximum operating condition while calculating the delay for another path based on minimum operating condition for setup or hold checks. For more information, see "On-Chip Variation (OCV) Timing Analysis Mode," in the *Innovus User Guide*.

If you read in one library and do not specify any timing analysis type, the software uses `single` by

default. If you read in two libraries and do not specify any timing analysis type, the software uses `best_case_worst_case` by default.

timing_aocv_analysis_mode

```
timing_aocv_analysis_mode
{launch_capture | clock_only | separate_data_clock | combine_launch_capture}
```

Default: `launch_capture`

Controls handling of AOCV analysis modes.

- `launch_capture`: AOCV analysis will be performed for both data and clock paths, and the AOCV derating factor will be calculated based on both clock and data path stages.
- `clock_only`: AOCV analysis will be performed for clock paths only, and the AOCV derating factor will be calculated based on the clock path stage depth only.
- `separate_data_clock`: The software calculates the AOCV stage count for clock paths and data paths separately. Both clock and data paths have related AOCV derating factors.
- `combine_launch_capture`: The software calculates AOCV stage count using combined depth for launch and capture paths. This is achieved by summation of path depth on the launch and capture paths.

timing_aocv_chip_size

```
timing_aocv_chip_size real_number
```

Default: Automatically calculated from SPEF file, if one is loaded.

Specifies the diagonal length of the chip, in microns.

The chip and core sizes specified with the `timing_aocv_chip_size` and `timing_aocv_core_size` attributes take priority over chip and core size values derived from the SPEF file, or through placement. In the case that there is no SPEF file loaded, or placement information available:

- If you specify `timing_aocv_chip_size` but not `timing_aocv_core_size`, the software considers the core size to be equal to the `timing_aocv_chip_size` value.
- If you specify `timing_aocv_core_size` but not `timing_aocv_chip_size`, the software ends the AOCV analysis and generates an error message.
- If both `timing_aocv_chip_size` and `timing_aocv_core_size` are not specified, the software ends

the AOCV analysis and generates an error message.

timing_aocv_core_size

`timing_aocv_core_size` *`real_number`*

Default: Automatically calculated from SPEF file, if one is loaded.

Specifies the diagonal length of the core area, in microns.

The chip and core sizes specified with the `timing_aocv_chip_size` and `timing_aocv_core_size` attributes take priority over chip and core size values derived from the SPEF file, or through placement. In the case that there is no SPEF file loaded, or placement information available:

- If you specify `timing_aocv_chip_size` but not `timing_aocv_core_size`, the software considers the core size to be equal to the `timing_aocv_chip_size` value.
- If you specify `timing_aocv_core_size` but not `timing_aocv_chip_size`, the software ends the AOCV analysis and generates an error message.
- If both `timing_aocv_chip_size` and `timing_aocv_core_size` are not specified, the software ends the AOCV analysis and generates an error message.

timing_aocv_derate_mode

`timing_aocv_derate_mode {aocv_multiplicative | aocv_additive}`

Default: `aocv_multiplicative`

Controls the AOCV derating mode.

When set to `aocv_multiplicative`, the derating factor will be calculated as AOCV derating * OCV derating, which is set using the `set_timing_derate` command.

When set to `aocv_additive`, the derating factor will be calculated as AOCV derating + OCV derating values.

When you use this attribute, the `report_timing` command shows the `total_derate` column in the timing report output, which allows you to view and cross-check the calculated total derate factor.

timing_aocv_stage_count_update_on_timing_reset

`timing_aocv_stage_count_update_on_timing_reset {true | false}`

Default: `false`

When set to `true`, the AOCV stage counts are re-calculated when timing is reset.

timing_apply_check_derate_to_external_output_delay

`timing_apply_check_derate_to_external_output_delay {true | false}`

Default: `false`

Applies check derate to output external delay.

The `set_timing_derate -cell_check` parameter by default does not apply check derate to output external delay (using the `set_output_delay` command). To enable this feature, you can set the `timing_apply_check_derate_to_external_output_delay` timing attribute to `true`. This attribute applies cell (check) derate to the top boundary port.

timing_apply_default_primary_input_assertion

`timing_apply_default_primary_input_assertion {true | false}`

Default: `true`

When set to `true`, primary input and bidirectional ports that do not have an explicit arrival time specified (through the `set_input_delay` constraint), are provided a default arrival time. The arrival is referenced to an internal default clock that is indicated by the `@` sign in various timing reports.

If a `set_input_delay` constraint is assigned to the input port, but the `-clock` parameter was not specified, the software assumes the constraint to be with reference to the default clock.

When set to `false`, inputs ports with no explicit `set_input_delay` are unconstrained for sequential timing analysis.

timing_apply_exceptions_to_data_check_related_pin

`timing_apply_exceptions_to_data_check_related_pin {true | false}`

Default: `true`

Controls if false path assertions which block the data path to the related (-from) reference pin of the data-to-data check disable the data-to-data check.

When set to `false`, the data-to-data checks remain valid when a false path assertion blocks the data path to the related (-from) pin.

When set to `true`, any false path assertion which blocks the data path to the related (-from) pin of the data-to-data check also causes the check to be disabled.

This attribute has no effect on the data-to-data checks with a reference clock, where an exception blocks the clock reference signal at the reference pin. The data-to-data check will always be disabled in this situation.

Examples

- In the following example, the data-to-data check is disabled:

  ```
  set timing_apply_exceptions_to_data_check_related_pin true
  set_data_check -from portA -to portB
  set_false_path -to portA
  ```

- In the following example, the data-to-data check is valid. This is the default situation:

  ```
  set timing_apply_exceptions_to_data_check_related_pin false
  set_data_check -from portA -to portB
  set_false_path -to portA
  ```

- In this case, the data check is always disabled and is not affected by this attributes' settings:

  ```
  set_data_check -from portA -to portB
  set_false_path -to [get_clocks ClkA]
  ```

  where the path to `portA` is clocked by `ClkA`.

timing_apply_setup_hold_exceptions_to_data_check_related_pin

`timing_apply_setup_hold_exceptions_to_data_check_related_pin {true | false}`

Default: `true`

Disables the early signal data-to-data paths, when false path is applied on the late reference path.

When set to `true`, the `set_false_path -setup` parameter settings on the reference path disables early signal d2d path, and vice versa. If both the `-setup` and `-hold` parameters are set, the signal early/late d2d path is disabled, irrespective of this attribute settings.

When set to `false`, the `set_false_path` exceptions set on d2d reference path do not affect data path slack. When the `set_false_path` command is set on the reference d2d path with only `-setup` (or only `-hold`) parameter, the signal path of d2d check is not affected by these exceptions.

timing_cap_unit

`timing_cap_unit` *`string`*

Default: " "

Specifies the capacitance units to be used for timing constraints and timing reports. The default unit for capacitance is inferred from the timing libraries. You can override this inference by explicitly setting the value to 1pf or 1ff. Setting this variable explicitly is the same as executing the `set_library_unit -cap` command to set the capacitance units.

timing_case_analysis_propagation

`timing_case_analysis_propagation {true | false}`

Default: `true`

Enables case analysis propagation.

When set to `true`, the software overrides sequential and ICG propagation behavior.

When set to `false`, the software disables case analysis propagation.

timing_case_analysis_for_sequential_propagation

`timing_case_analysis_for_sequential_propagation {true | false}`

Default: `false`

When set to `false`, the constants will stop at the sequential elements.

When set to `true`, the software propagates constants through sequential elements. You can set `timing_case_analysis_for_icg_propagation` attribute to `require_seq_prop` to allow propagation of constants through ICG cells.

When you change the setting of this attribute, each time a complete timing update is required and all the incremental updates are abandoned. If the outputs evaluate to a constant, the check arcs and delay arcs are disabled regardless of the setting.

timing_case_analysis_for_icg_propagation

`timing_case_analysis_for_icg_propagation {false | require_seq_prop | always}`

Default: `false`

Determines in which conditions the constants are propagated through the integrated clock gating

(ICG) cells.

The available options are described below:

Option	Description
`always`	Enables constant propagation through ICGs, regardless of the value of the `timing_case_analysis_for_sequential_propagation` global variable, or the value of `timing_case_analysis_for_sequential_propagation` setting. This allows the software to have constant propagation for ICG cells, even when this is disabled for other sequential devices.
`false`	Disables constant propagation through ICG cells.
`require_seq_prop`	When the `require_seq_prop` option is used, the software will treat ICG cells like other cells in the design from the perspective of constant propagation. This means that the `timing_case_analysis_for_sequential_propagation` global variable or `timing_case_analysis_for_sequential_propagation` setting will act as a master switch and will control constant propagation through ICGs also.

timing_check_timing_report_all_checks

`timing_check_timing_report_all_checks {true | false}`

Default: `false`

Enables reporting of all the checks performed by the `check_timing` command.

When set to `true`, the `check_timing` command reports all the checks performed, even if there is no violation(s) for the corresponding check type.

When set to `false`, only the check types that have any violations will be reported.

timing_check_timing_signal_level_high_to_low_threshold

`timing_check_timing_signal_level_high_to_low_threshold` d*ouble*

Default: `0`

Sets the threshold value for warning the signal level difference between driver and receiver voltages, when the driver is at a higher voltage than the receiver voltage.

The software shows the signal level warnings (`check_timingsignal_level`) that report the net driver/receiver pin pairs for which the pin voltage does not match. The signal level warning messages can be filtered based on these threshold values.

timing_check_timing_signal_level_low_to_high_threshold

`timing_check_timing_signal_level_low_to_high_threshold` d*ouble*

Default: `0`

Sets the threshold value for warning the signal level difference between driver and receiver voltages, when the driver is at a lower voltage than the receiver voltage.

The software shows the signal level warnings (`check_timingsignal_level`) that report the net driver/receiver pin pairs for which the pin voltage does not match. The signal level warning messages can be filtered based on these threshold values.

timing_clock_phase_propagation

`timing_clock_phase_propagation {positive | negative | both}`

Default: `both`

Allows you to select the appropriate clock phases at register clock pins when both the positive and negative phases of the same clock signal are seen in the clock network. For example, a register whose clock pin is fed by a two-input XOR gate where one input is a clock port and the other input is a data port. When the value of the data signal of the XOR gate is unknown, the XOR gate can see both positive and negative phases of the clock signal.

Option	Description
`both`	Selects both positive and negative clock phases at the register clock pins.
`positive`	Selects positive clock phases.
`negative`	Selects negative clock phases.

timing_clock_source_paths_unconstrained_mark_clock_used_as_data

`timing_clock_source_paths_unconstrained_mark_clock_used_as_data {true | false}`

Default: `true`

The `is_clock_used_as_data` property (by using the `get_property` command) returns a value of `true` for unconstrained clock source network.

timing_clock_source_use_driving_cell

```
timing_clock_source_use_driving_cell{true | false}
```

Default: `true`

When set to `false`, delay calculation in propagated clock mode will use an idealized driver model at the `create_clock` root pin or port when it calculates delays for the first stage of the clock tree. The idealized model is a ramp input with the transition time set by the system default transition, which is normally 0ns.

When this variable is set to `true`, the drive characteristics of the cell output pin that `create_clock` is asserted on will be used to calculate the first stage clock tree delays and slews. The input slew used for this driver model is whatever slew has been propagated to the cell's input pins from upstream logic. The arrival time of the clock at the root pin is not modified in any way.

If you wish to specify the input slews to be used for the clock driver cell, that is, not use the upstream propagated slews - you can use the `set_annotated_transition` constraint on the cell's input pins to control this. Setting an input transition of 0ns on the input pins can be useful when trying to correlate analysis results with other applications which do not use the full propagated transition times.

timing_clock_uncertainty_from_to_precedence

```
timing_clock_uncertainty_from_to_precedence {true | false}
```

Default: `false`

Sets the clock uncertainty precedence.

When set to `true`, the software gives higher precedence to clock-to-clock uncertainty.

timing_collection_all_fanin_fanout_traversal_mode

```
timing_collection_all_fanin_fanout_traversal_mode {true | false}
```

Default: `flat`

Range: `{hierarchical flat}`

Controls how the software processes netlists during `all_fanin`/`all_fanout` command processing.

timing_collection_result_display_limit

`timing_collection_result_display_limit` *integer*

Default: `100`

Limits the number of objects of a collection to be displayed in the output report.

When set to an integer value, all the commands, such as, `get_cells`, `add_to_collection`, and `query_objects` commands, that display collection objects in the output report will be impacted.

When set to `-1`, all the elements in the collection will be displayed in the output.

timing_collection_variable_assignment_compatibility

`timing_collection_variable_assignment_compatibility {true | false}`

Default: `true`

Disables printing the names of the output of `get_*` commands.

When set to `false`, the software prints the `get_*` command output on the shell. This attribute should be used for debugging purposes only.

timing_constraint_disable_min_max_input_delay_worst_casing

`timing_constraint_disable_min_max_input_delay_worst_casing {true | false}`

Default: `false`

Disables the override of the maximum value (set using the `set_input_delay -max` parameter) from the specified `-min` value.

timing_constraint_enable_detailed_report_invalid_begin_end_points

`timing_constraint_enable_detailed_report_invalid_begin_end_points{true | false}`

Default: `false`

Enables detailed reporting of invalid begin/end points specified with the `set_false_path`/`set_multicycle_path` constraints.

When set to `false`, the software checks only the structural end points.

timing_constraint_enable_drv_limit_override

`timing_constraint_enable_drv_limit_override {true | false}`

Default: `true`

Controls support of the DRV override flow.

When set to `true`, the software allows you to override the limits from other sources, such as, library or constraints. To use this feature, you can specify the `-override` parameter with the following commands:

- `set_max_capacitance`
- `set_max_transition`
- `set_max_fanout`
- `set_min_capacitance`
- `set_min_transition`
- `set_min_fanout`

When specified, the software forces a specific constraint value for the specified list of instance pins, ports, or library cell pins to override the maximum/minimum values set prior to these constraint settings.

timing_constraint_enable_logging

`timing_constraint_enable_logging {true | false}`

Default: `false`

Enables dumping of constraint commands in the log file.

Note: The impact of this attribute is visible only when the constraints are loaded using the `read_sdc` command or the view definition file.

timing_constraint_enable_report_invalid_begin_end_points

`timing_constraint_enable_report_invalid_begin_end_points {true | false}`

Default: `false`

Enables reporting of invalid begin points/end points specified with the `set_false_path` or `set_multicycle_path`constraint.

timing_constraint_enable_search_path

`timing_constraint_enable_search_path {true | false}`

Default: `false`

Enables searching of constraint files in search_path directory variable.

timing_constraint_enable_separate_multicycle_data_checks

`timing_constraint_enable_separate_multicycle_data_checks {true | false}`

Default: `false`

Controls multicycle path constraint application on data checks.

timing_constraint_load_minimal_set_for_automated_eco

`timing_constraint_load_minimal_set_for_automated_eco {true | false}`

Default: `false`

Skips loading the SDC constraints that are not required, when the `opt_design` command is used. The timing data is obtained from the ECO timing database - this global variable reduces the TAT for designs with complex constraint mode files.

timing_constraint_path_delay_exclude_check_delay_from_ignore_clock_latency

`timing_constraint_path_delay_exclude_check_delay_from_ignore_clock_latency {true | false}`

Default: `false`

Excludes check arc delays from slack computation.

When set to `true`, the software excludes check arc delays from slack computation for paths constrained by the `set_max_delay`/`set_min_delay`-`ignore_clock_latency` command.

timing_constraint_path_delay_exclude_io_delay_from_ignore_clock_latency

`timing_constraint_path_delay_exclude_io_delay_from_ignore_clock_latency {true | false}`

Default: `false`

Excludes the input/output delays from slack computation.

When set to `true`, the software excludes the input/output delays from slack computation for paths constrained by the `set_max_delay`/`set_min_delay`-`ignore_clock_latency` command.

timing_constraint_path_delay_exclude_unconstrained_endpoints

`timing_constraint_path_delay_exclude_unconstrained_endpoints {true | false}`

Default: `false`

When path delay exceptions are asserted (using the `set_min_delay` or `set_max_delay` commands) with a complete `-from` and `-to` specification, the exception can be applied between any two data path pins. If the constraint assertion includes only a -from and/or -through argument, then a set of valid endpoints must be established to define the constrained paths. This attribute allows you to configure the set of valid endpoints (inferred -to pins) that are selected.

When set to `false`, the set of allowed endpoints is simply all of the endpoints in the combinational fanout of the `-from` and/or `-through` arguments - essentially, wherever the path tracing stops. The path delay constraint will be applied as if there were a -to argument specified for each endpoint.

When set to `true`, the software uses different rules to determine the valid set of inferred -to endpoints. Only constrained pins in the fanout of -from and/or -through arguments are considered as valid -to endpoints for the constraint. A constrained endpoint will have some type of timing check - such as Setup or Hold already in force on that pin. Additionally, each transition at the endpoints will be examined to determine if each is individually constrained. The path exception will only be applied to the edges that are constrained. As an example, if a register preset or a clear pin was a potential endpoint, then only the edge constrained by the recovery or removal check will have the path delay applied. To apply the path delay constraint to unconstrained endpoints while in this mode, you must specify them explicitly with the `-to`, `-rise_to`, or `-fall_to` parameters of the `set_min_delay` and `set_max_delay` constraints.

timing_constraint_path_delay_include_clock_pin_endpoints

`timing_constraint_path_delay_include_clock_pin_endpoints {true | false}`

Default: `false`

Considers register clock pins as end-points for path delay constraints applied upstream.

When set to `true`, the `set_max_delay` command considers unconstrained paths ending at the register clock pins.

timing_constraint_warn_for_timing_derate_exceeding_max_limit

`timing_constraint_warn_for_timing_derate_exceeding_max_limit` *`double`*

Default: `100`

Value Range: 0 to 100

Sets a limit on derate values. If a larger derate value is applied, the derates are still processed. However, the software issues a warning.

Note: It is recommended to specify derate values within the acceptable value range, because large derate values may cause errors during processing.

timing_continue_on_error

`timing_continue_on_error {true | false}`

Default: false

When set to `true`, the software continues to process if an error occurs when any constraints are used, or during timing analysis, or when a library is read. This attribute does not have any effect on other commands, except constraints or commands related to timing analysis.

When set to `false`, if an error occurs, during timing analysis, library reading, or defined constraints, the software stops processing and exits.

timing_constraint_update_io_latency_averaging_mode

`timing_constraint_update_io_latency_averaging_mode {rise_fall | rise_only}`

Default: `rise_fall`

Controls the analysis to update IO latency for both rise and fall, or for rise only. You can set one of the options - `rise_fall`, `rise_only`.

timing_context_clock_mapping_percentage_tolerance

`timing_context_clock_mapping_percentage_tolerance {true | false}`

Default : `false`

Controls the tolerance value in clock mapping.

timing_context_clock_mapping_check_waveform

`timing_context_clock_mapping_check_waveform {true | false}`

Default : `true`

Controls the waveform checking.

timing_cppr_enable_mismatch_transition_mode

`timing_cppr_enable_mismatch_transition_mode {true | false}`

Default: `true`

Enables CPPR compatibility mode when transitions mismatch at the common pin.

This attribute controls the CPPR adjustment calculations that are computed at the common pin when transitions for the launch and capture paths mismatch at the common pin. The following case scenarios are possible:

- The launch and capture clocks at the common pin are different but match the launch and capture clocks reaching the endpoint, or, launch and capture clock at the common pin are same but different from the launch and capture clocks reaching the end point.
- The launch and capture clocks at the common pin are same and match the launch and capture clocks reaching the endpoint.

The following examples show the behavior for CPPR computations when this attribute is set to `true`:

- Consider a launch path with clock CK1, late rise transition and a capture path with clock GCK1 (related to CK1), early fall transition. The CPPR adjustment is computed by considering the late fall from clock CK1 and early rise from clock GCK1.

  When set to `false` (default), the software considers late fall from clock GCK1 and early rise from clock CK1.
- Consider a launch path with clock CK1, late rise transition and a capture path with clock CK1,

early fall transition. The CPPR adjustment is computed by considering the late fall from the launch phase of CK1 and early rise from the capture phase of CK1.

When set to `false`, the software considers the late fall from capture phase of CK1 and early rise from launch phase of CK1.

The CPPR adjustment is the minimum of the two CPPR adjustments computed for the rise and fall transitions.

All the other combinations of the launch and capture transitions at the common pin for late and early paths will show similar changes to the CPPR adjustment computation - when set to `true`.

timing_cppr_opposite_edge_mean_scale_factor

`timing_cppr_opposite_edge_mean_scale_factor` *`double`*

Default: 1.0

Value Range: 0 to 1

Controls the mean delay component for CPPR credit for opposite transition edges at a common pin.

When this attribute is specified, the software provides a finer level configuration of the CPPR credit for reducing Common Path Pessimism Removal (CPPR) in timing analysis. The software allows a certain amount of CPPR credit in the case of opposite edge transitions to be scaled to a user-specified value, that is between zero credit or full credit provided by the existing controls.

You can use `timing_cppr_opposite_edge_sigma_scale_factor` attribute for adjustment of the sigma component of the CPPR credit. In SOCV analysis mode, you can use both of these attributes to separately control the adjustment of the mean and sigma components of the CPPR credit. In regular static-timing analysis (STA) mode (without SOCV), only the mean scale factor is considered for CPPR adjustment scaling.

timing_cppr_opposite_edge_sigma_scale_factor

`timing_cppr_opposite_edge_sigma_scale_factor` *`double`*

Default: 1.0

Value Range: 0 to 1

Controls sigma delay component for CPPR credit for opposite transition edges at common pin.

timing_cppr_opposite_edge_sigma_scale_factor_net

`timing_cppr_opposite_edge_sigma_scale_factor_net` *double*

Default: `1`

Controls the net sigma delay component for CPPR credit for opposite transition edges at the common pin.

The scale factors are applicable only when the common pin has mismatched transitions for the launch and capture paths.

timing_cppr_opposite_edge_sigma_scale_factor_cell

`timing_cppr_opposite_edge_sigma_scale_factor_cell` *double*

Default: `1`

Controls the cell sigma delay component for CPPR credit for opposite transition edges at the common pin.

The scale factors are applicable only when the common pin has mismatched transitions for the launch and capture paths.

timing_cppr_propagate_thru_latches

`timing_cppr_propagate_thru_latches {true | false}`

Default: `false`

When set to `true`, the CPPR of a timing path crossing a latch (or latches), for example, ff1 -> lat1 -> ff2, is calculated using the common point of the origin device (ff1) and the capture device at the end of the path (ff2).

When set to `false`, the segment-based CPPR calculation is performed, considering the common point of latch (lat1) and the capture device (ff2). This CPPR setting is considered only when the latch thru analysis mode is enabled and a latch thru path exists.

timing_cppr_remove_clock_to_data_pessimism

`timing_cppr_remove_clock_to_data_pessimism {true | false}`

Default: `false`

When set to true, removes clock re-convergence pessimism (CRP) for clock source paths.

timing_cppr_self_loop_mode

`timing_cppr_self_loop_mode {true | false}`

Default: `true`

When set to `true` in case of self-loop paths, computes CPPR adjustment by taking the difference between early and late clock arrival time of the common point; the clock pin of the flop. A self-loop path is a path that goes from a clock pin of a flop and ends at the data pin of the same flop. However detecting self-loop paths requires runtime, and is design dependant.
When set to `false`, does not detect self-loop paths. In this case, the software computes CPPR adjustment by using the common point as the output pin driving the clock pin of the flop. This might result in better runtime, but less accurate results. The loss of accuracy is the difference between early and late delay of the interconnect portion driving the clock pin.

timing_cppr_skip_clock_reconvergence

`timing_cppr_skip_clock_reconvergence {true | false}`

Default: `false`

Specifies the branch point to use for computing clock path pessimism removal (CPPR) adjustment when there is reconvergence in the clock tree.

When set to `false`, the software uses the branch point closest to the register clock pins where the clocks reconverge. When set to `true`, the software uses the farthest branch point from the register clock pins (that is, the branch point closest to the clock root pin where the clocks diverge).

timing_cppr_skip_clock_reconvergence_for_unmatched_clocks

`timing_cppr_skip_clock_reconvergence_for_unmatched_clocks {true | false}`

Default: `false`

Allows the software to skip re-convergence of unmatched clocks.

When set to `true`, the software uses the farthest branch point from the register clock pins when the launching and capturing clocks are different - the re-converge points are ignored and not searched.

timing_cppr_threshold_ps

`timing_cppr_threshold_ps` *`float`*

Default: `20` ps

Specifies the maximum amount of pessimism that clock path pessimism removal (CPPR) analysis is allowed to leave in the path. The unit is pico seconds. Setting this attribute to a specified value means that all the paths might be reported without having their pessimism removed. This attribute can be used to speed up CPPR analysis run time. The larger the value of the attribute, the faster the run time.

timing_cppr_transition_sense

```
timing_cppr_transition_sense {normal | same_transition | same_transition_expanded}
```

Default: `normal`

Specifies the transition sense of the launching and capturing clocks at the common node, to calculate clock path pessimism removal (CPPR).

- `normal`: The CPPR analysis computes the CPPR value even if the launching and capturing clock transitions at the common node are in opposite directions. If the transition sense at the common node are in opposite directions, the pessimism is the minimum of the difference between late and early arrival times for rise and fall clock transitions.
- `same_transition`: The CPPR analysis computes the CPPR value only if the launching clock transition is same as the capturing clock transition at the common node. If the transitions are different, the CPPR value is zero.
- `same_transition_expanded`: The CPPR analysis computes the CPPR value only if the launching clock transition is the same as the capturing clock transition at the common node. If the transitions are different, the software searches until it finds a common point where the transitions are the same, and calculates the CPPR value from that point.

timing_create_clock_default_propagated

```
timing_create_clock_default_propagated {true | false}
```

Default: `false`

Enables clocks to be created in propagated mode.

By default, all clocks, created using `create_clock` and `create_generated_clock` commands, are in ideal mode.

When the attribute is set to `true`, all subsequently created regular clocks, derived clocks, and library generated clocks are in propagated mode.

There is no effect on the functionality of `set_propagated_clock` and `set_clock_latency` commands.

timing_default_opcond_per_lib

```
timing_default_opcond_per_lib {true | false}
```

Default: `true`

Determines the default operating condition behavior when there is no explicit user operating condition.

When set to `true`, each timing library uses its own internally-defined default operating condition specification.

When set to `false`, the default operating condition of the master library (that is, the first library in the library search list) is used to set the PVT operation point for the entire design. All timing libraries are accessed and derated to that operating point.

timing_defer_mmmc_obj_updates

```
timing_defer_mmmc_obj_updates {true | false}
```

Default: `false`

By default, the multi-mode multi-corner (MMMC) update commands, `update_constraint_mode` and `update_analysis_view`, actively load new data upon execution. Since all the timing information is reloaded with each update command, executing a sequence of updates can result in multiple reloads and a subsequent degradation of performance and turn-around-time.

When set to `true`, the software allows a sequence of MMMC updates to be accumulated before new data is loaded and analyzed.

If the `set_analysis_view -update_timing` parameter is specified, the software indicates that the MMMC object updates are complete and new configuration can be loaded and analyzed.

For example:

```
update_constraint_mode -name ioTiming -sdc_files {postCtsIO.sdc}

update_constraint_mode -name coreTiming -sdc_files {postCtsCore.sdc}

set_analysis_view -update_timing
```

timing_derate_aocv_dynamic_delays

```
timing_derate_aocv_dynamic_delays {true | false}
```

Default: `true`

Controls AOCV derating of SI-induced delays.

When set to `false`, AOCV derates will not be applied on dynamic, SI-induced delays, that is, a value of `false` ensures AOCV derating is only applied to the static component of the delay.

When set to `true`, AOCV derating is applied to both static and dynamic arc delay components.

timing_derate_aocv_reference_point

```
timing_derate_aocv_reference_point {true | false}
```

Default: `1`

Allows specification of reference point for AOCV factors. As per the current AOCV library definitions, the AOCV reference point is always expected to be set at `1.0`.

timing_derate_dynamic_compatibility

```
timing_derate_dynamic_compatibility {true | false}
```

Default: `1`

Controls how the `set_timing_derate` command factors that are not specified, using either `-static` or `-dynamic` options, are applied to the delay.

When this attribute is set to `1`, the behavior will be compatible with previous releases. In this mode, the static and dynamic components will be summed up before applying derating.

When set to 0, the static and dynamic components will be derated separately and then combined. A setting of 0 is equivalent to using separate derate assertions with the `set_timing_derate -dynamic` and `-static` options explicitly specified.

timing_derate_ocv_reference_point

```
timing_derate_ocv_reference_point {true | false}
```

Default: `1`

Allows specification of reference point for OCV factors. As per the current OCV library definitions, the OCV reference point is always expected to be set at `1.0`.

timing_derate_negative_delay_backward_compatibility

```
timing_derate_negative_delay_backward_compatibility {true | false}
```

Default: `true`

Controls the effective derate calculation for negative delays. When effective derate is requested for negative delays, the value will be converted according to the negative delay scaling rule, e.g., 2 - positiveVal.

timing_derate_spatial_distance_unit

```
timing_derate_spatial_distance_unit {default | 1um | 1nm}
```

Default: `default`

Specifies distance units in the SOCV and AOCV library formats.

This attribute supports 3 distance units - `default`, `1um`, and `1nm`. The default value is the default units that are derived from the current system units for distance (um/nm overrides the default).

Note: The `timing_derate_spatial_distance_unit` attribute should be set before a design is loaded.

timing_derate_incremental_adjust_additive_mode

```
timing_derate_incremental_adjust_additive_mode {true | false}
```

Default: `false`

Allows the software to accumulate incremental derate values instead of overwriting them.

When set to `false`, the successive application of incremental derating will overwrite any existing derate value with the new value.

When set to `true`, the successive application of the incremental derates will be accumulated into a final derate value.

timing_disable_bus_contention_check

```
timing_disable_bus_contention_check {true | false}
```

Default: `false`

When set to `true`, disables propagation of maximum delay along three state disable timing arcs and minimum delay along three state enable arcs. If you know that bus contention conditions do not occur in your design, disable these checks by setting these attributes to `true`.

Checks for setup and hold violations in three-state bus designs. Checks the worst delay path to

ensure that the latched signals are stable and not unknown values - X. Ensures proper timing checks for bus contention.

timing_disable_clock_period_checks

```
timing_disable_clock_period_checks {true | false}
```

Default: `false` (performs checks)

When set to `true`, disables timing model clock period checks during timing analysis.

timing_disable_constant_propagation_for_sequential_cells

```
timing_disable_constant_propagation_for_sequential_cells {true | false}
```

Default: `false`

Controls constant propagation across sequential cells.

When set to `true`, the software disables constant propagation across sequential cells.

When set to `false`, the software enables constant propagation across sequential cells.

timing_disable_drv_report_on_constant_nets

```
timing_disable_drv_report_on_constant_nets {true | false}
```

Default: `false`

Disables DRV checks for nets that are disabled by constant propagation:
When the attribute is set to true, the report_constraint command does not report DRV violations for constant nets.

timing_disable_floating_bus_check

```
timing_disable_floating_bus_check {true | false}
```

Default: `false`

When set to `true`, disables propagation of minimum delay through three state disable timing arcs and maximum delay through three state enable arcs. These checks are only valid during floating bus conditions.

timing_disable_genclk_combinational_blocking

```
timing_disable_genclk_combinational_blocking
{true | false}
```

Default: `false`

Disables blocking of the latency path for a generated clock.

When set to `true`, allows the latency path to traverse through the upstream generated clock.

When set to `false`, if two back-to-back generated clocks are on the same path, and if the downstream generated clock is set using the `create_generated_clock -combinational` parameter, then the latency path for this generated clock is blocked by the upstream clock. The downstream generated clock is created with zero latency.

timing_disable_inferred_clock_gating_checks

```
timing_disable_inferred_clock_gating_checks {true | false}
```

Default: `false` (performs checks)

When set to `true`, disables clock gating checks that are inferred on combinational elements in the clock path. Explicit clock gating checks that are described in the timing library are not affected by this attribute.

timing_disable_inout_output_side_timing_checks

```
timing_disable_inout_output_side_timing_checks {true | false}
```

Default: `true`

Disables timing checks on both the input and output sides of the bidirectional pin.

When set to `false`, the timing checks on both the input and output sides of the bidirectional pin are analyzed.

timing_disable_internal_inout_cell_paths

```
timing_disable_internal_inout_cell_paths {true | false}
```

Default: `true`

When set to `false`, enables internal bidirectional feedback paths that are completely contained in one

instance.

timing_disable_internal_inout_net_arcs

`timing_disable_internal_inout_net_arcs {true | false}`

Default: `true`

When set to `true`, this attribute disables internal bidirectional feedback paths that span multiple instances.

When set to `false`, this attribute enables internal bidirectional feedback paths, which are feedback paths that you might not want to use during analysis and optimization.

This attribute applies only to paths that span multiple instances. The attribute has no impact on internal feedback paths that are completely contained in one instance.

timing_disable_lib_pulse_width_checks

`timing_disable_lib_pulse_width_checks {true | false}`

Default: `false` (performs checks)

When set to `true`, disables timing model pulse width checks during timing analysis.

timing_disable_library_data_to_data_checks

`timing_disable_library_data_to_data_checks {true | false}`

Default: `false`

When set to `true`, this attribute disables data-to-data checks that are coded in the library as non-sequential timing checks. Data-to-data checks are distinguished from other non-sequential timing checks because both the reference pin and constrained pin side of the timing check are data pins rather than clock pins. You should use the `timing_disable_non_sequential_checks` attribute to control other types of non-sequential checks where the reference pin is a clock pin.

timing_disable_library_tieoffs

`timing_disable_library_tieoffs {true | false}`

Default: `false`

Disables constant functions in a library.

When set to `false`, the software considers constant propagation from library tie high/low cells prior to those from logical netlist and constraints. There may be cases where programmable tie cells are used whose value can be changed later in the flow, assuming the programmable tie cells are modelled with constant output functions.

When set to `true`, the library constants are not propagated during timing analysis or timing optimization.

timing_disable_netlist_constants

```
timing_disable_netlist_constants {true | false}
```

Default: `false`

When set to `true`, ignores constants defined in the Verilog netlist. This includes all uses of the in-place constant notation, such as `1'b0`, and the use of power and ground nets. Constants from `set_case_analysis` assertions and from tie-hi and tie-lo cells are still recognized.

When set to `false`, the software recognizes all constants.

timing_disable_nochange_checks

```
timing_disable_nochange_checks {true | false}
```

Default: `false` (performs checks)

When set to `true`, disables no change timing model checks during timing analysis.

timing_disable_non_sequential_checks

```
timing_disable_non_sequential_checks {true | false}
```

Default: `false`

When set to `true`, disables the timing arcs between any data-to-clock or clock-to-clock checks.

This attribute only affects non-sequential checks where the reference pin of the timing check is a clock pin. Data-to-data checks - where the reference pin of the timing check is a data pin - are not affected by this attribute. To disable data-to-data checks you must use the `timing_disable_library_data_to_data_checks` attribute.

timing_disable_output_as_clock_port

`timing_disable_output_as_clock_port{true | false}`

Default: `false`

Allows a clock path to be considered as a data path.

When set to `true`, this attribute causes a clock source path, leading to an output or bidi port, to be treated as a data path, irrespective of any constraint like `set_output_delay`, `set_data_check`, or `set_max_delay` asserted on that port.

timing_disable_pulse_width_same_edge_si_cppr_mode

`timing_disable_pulse_width_same_edge_si_cppr_mode {true | false}`

Default: `true`

Enables SI CPPR credit up to the last divergent pin for pulse width checks for zero width pulse clocks. By default, the software credits minimum of rise and fall transition CPPR up to the common pin as the transitions mismatch at the common pin.

When set to `true`, the software computes base delay CPPR up to the common pin as CPPR credit for the path. The software then finds an alternate common timing pin in the clock network where the transition for launch and capture match. If a match is found, the software includes the SI delay up to this pin in the CPPR credit as well.

timing_disable_report_header_info

`timing_disable_report_header_info {true | false}`

Default: `false`

Controls whether timing reports are generated using a common report header.

When set to `false`, the software generates timing reports with a report header that includes the following information:

- Software version number
- Platform on which the software is being run
- Time when the report was generated
- Command used to generate the report

Any report-specific banner information is included in the same header box, but listed after the common

header information.

When set to `true`, the software generates backward-compatible timing reports without the additional header information.

timing_disable_retime_clock_path_slew_propagation

```
timing_disable_retime_clock_path_slew_propagation {true | false}
```

Default: `true`

When set to `true`, path based analysis is performed on data paths only and the clock segments are not considered.

When set to `false`, analysis is performed on both data and clock paths.

timing_disable_sdf_retain_arc_merging

```
timing_disable_sdf_retain_arc_merging {true | false}
```

Default: `false`

Controls handling of retain arcs in SDF2.1.

When set to `false`, the Retain information is merged into SDF 2.1, as Retain represents a valid minimum path timing for a cell.

When set to `true`, the Retain arc merging will be disabled.

timing_disable_skew_checks

```
timing_disable_skew_checks {true | false}
```

Default: `false`

When set to `true`, disables library skew checks for timing analysis.

timing_disable_test_signal_arc

```
timing_disable_test_signal_arc {true | false}
```

Default: `false`

When set to `true`, timing analysis will not analyze the signal arcs coming from or going to the test pin. Such pins are marked in the timing library with either the `test_scan_in`, the `test_scan_enable` attribute

for the input pin, the `test_scan_clock` attribute for the input clock pin, or the `test_output_only` attribute for the output pin.

The following are the equivalent Liberty and TLF pin attributes:

Type	Liberty	TLF
enable	`test_scan_enable`	`SCAN_PINTYPE` (ENABLE)
input	`test_scan_in`	`SCAN_PINTYPE` (INPUT)
input clock	`test_scan_clock`	`SCAN_PINTYPE` (CLOCK)
output	`test_output_only`	`SCAN_PINTYPE` (OUTPUT)

timing_disable_timing_model_latch_inferencing

`timing_disable_timing_model_latch_inferencing {true | false}`

Default: `true`

Controls whether latch behavior is inferred for cell descriptions tagged with the Liberty `timing_model_type` attribute, including all values: `abstracted`, `extracted`, or `qtm`.

When set to `false`, the software infers latch behavior from the timing check and timing delay arc relationships in the model's description.

When set to `true`, latch inferencing does not happen for cells with the `timing_model_type` attribute. Explicit Liberty latch modelling constructs must be coded into the description in order to model latch behavior.

This attribute does not affect Liberty timing model cells that do not contain the `timing_model_type` attribute. Standard cell core libraries generally do not have this attribute specified; therefore, they typically are not be affected by this attribute setting.

timing_disable_tristate_disable_arcs

`timing_disable_tristate_disable_arcs {true | false}`

Default: `false`

When set to `true`, disables all 0/1->Z transitions of tristate arcs during timing analysis. By default, all tristate arcs are enabled.

timing_disable_user_data_to_data_checks

`timing_disable_user_data_to_data_checks {true | false}`

Default: `false`

When set to `true`, disables data to data checks that are created by the `set_data_check` command. This attribute does not affect non-sequential timing checks described in the timing library.

timing_driving_cell_override_library

`timing_driving_cell_override_library {true | false}`

Default: `false`

Controls the selection of library cells specified using the `set_driving_cell`command.

When set to `true`, a driving cell will be picked up from the Max Library for max drive cell adjustment and Min Library for min drive cell adjustment, irrespective of the library specified with the `set_driving_cell -library` parameter.

When set to `false`, the libraries are selected as specified with the `set_driving_cell` command.

timing_enable_aocv_slack_based

`timing_enable_aocv_slack_based {0 | 1}`

Default: 0

Enables AOCV analysis using slack-based method.

When set to `true`, the software will try to remove excess pessimism from the graph-based analysis (GBA) AOCV stage counts. The paths to endpoints, which are under the specified slack criteria (default 0.0 ns), are considered when computing stage counts. This allows the stage counts and subsequent AOCV derates to be aligned with the failing paths – and not unnecessarily punished by derates coming from related short paths that meet timing requirements.

Note: A `true` setting is recommended when performing Setup optimizations in Innovus – where over pessimism can cause sub-optimal results.

timing_enable_case_analysis_conflict_warning

`timing_enable_case_analysis_conflict_warning {true | false}`

Default: `false`

Displays a warning when propagated and asserted constant values are not the same when a full timing update is performed.

When set to `true`, a report file, named `CTE_constant_mismatch.rpt`, is generated that contains details of pins and the reason for constant mismatch. This report is over-written each time a full timing update takes place.

timing_enable_clock_to_clock_clock_gating_check

```
timing_enable_clock_to_clock_clock_gating_check {true | false}
```

Default: `false`

Enables clock-to-clock clock gating checks.

When set to `true`, the software performs clock to clock gating checks at gating elements. For clock to clock gating checks, the frequency of a clock arriving at the reference side of a check should be equal to or greater than the frequency of the clock arriving at the signal side of the check.

timing_enable_clock_phase_based_rise_fall_derating

```
timing_enable_clock_phase_based_rise_fall_derating {true | false}
```

Default: `false`

Determines the interpretation of edge-specific derating factors for clock paths, specified using the `set_timing_derate -rise/-fall` parameters.

When set to `false`, the `set_timing_derate -rise/-fall` parameter refer to the direction of the transition from the perspective of each individual output pin along the clock path.

When set to `true`, the `-rise/-fall` parameter refers to which edge (or phase) of the clock is triggering the transition as viewed from the source of the clock where the clock waveform is defined.

timing_enable_data_through_clock_gating

```
timing_enable_data_through_clock_gating {true | false}
```

Default: `true`

When set to `false`, blocks signals arriving on the enable of the clock-gating check.

When set to `true`, allows both data and clock signals at the enable pin of a clock-gating check to propagate forward past the check.

timing_enable_derating_for_pulse_width_checks

`timing_enable_derating_for_pulse_width_checks {true | false}`

Default: `false`

Controls whether the cell check derates, set by `set_timing_derate -cell_check` command, can be applied for pulse width checks.

When set to `true`, the pulse width checks honor cell check derate factors.

timing_enable_early_late_data_slews_for_setuphold_mode_checks

`timing_enable_early_late_data_slews_for_setuphold_mode_checks {true | false}`

Default: `false`

Controls calculated delays.

When set to `true`, the software allows use of both the minimum and maximum data slews for setup and hold checks. By default, the minimum or maximum slew values for clock pins are considered, and for data pins only the maximum slew value is considered while calculating the setup/hold timing check values.

This attribute enables an eight-way timing check for:

- Early/late libraries
- Clock/data pins
- Min/max slews

You can use this attribute in simultaneous setup or hold mode only. To enable this mode, you need to set `timing_enable_simultaneous_setup_hold_mode` to `true`. In this mode, you can use the `report_delay_calculation` command (`-from` and `-to` parameters) to report all the setup/hold check values corresponding to minimum/maximum combination of clock and data slews.

timing_enable_genclk_divide_by_inherit_parent_duty_cycle

`timing_enable_genclk_divide_by_inherit_parent_duty_cycle {true | false}`

Default: `false`

Controls inheritance of duty cycle from master clock for generated clocks with divide by option.

timing_enable_genclk_source_path_register_limit

`timing_enable_genclk_source_path_register_limit {true | false}`

Default: `false`

Limits generated clock source latency path to traverse across one register only.

When set to `false`, the software allows master clock latency path to traverse across any number of sequential devices to reach the generated clock target pin.

When set to `true`, the software restricts the master clock latency path for a generated clock to traverse exactly one sequential device.

timing_enable_generated_clock_edge_based_source_latency

`timing_enable_generated_clock_edge_based_source_latency {true | false}`

Default: `true`

Controls how the software chooses generated clock source latency paths. Certain specifications of the `create_generated_clock`constraint imply a cause-and-effect relationship between the edges of the master clock and those of the generated clock.

When set to `true`, the software checks whether a path being analyzed has the correct cause-effect relationship consistent with the specified constraint. Paths that are found to be inconsistent are ignored. Only the paths that are in agreement with the required edge relationship are considered for source latency. If no such path exists, the software uses a source latency of `0` ns. This selection is done independently for each edge of the generated clock; therefore the result might be different for each edge.

The edge-to-edge sufficiency checks are performed assuming a Rise-to-Rise, Rise-to-Fall relationship between the master clock and the generated clock with an even number `-divide_by` factor. For the odd `-divide_by` factors, the sufficiency check is performed assuming a Rise-to-Rise, Fall-to-Fall relationship. The sufficiency checking is not performed for `-multiply_by` options.

When set to `false`, the software does not check paths for the correct cause-effect relationship.

timing_enable_get_obj_escaped_name_backward_compatible

`timing_enable_get_obj_escaped_name_backward_compatible {true | false}`

Default: `false`

Enables printing of escape characters in the `get_object_name` command output.

timing_enable_get_objects_regexp_compatibility

```
timing_enable_get_objects_regexp_compatibility {true | false}
```

Default: `false`

Defines the pattern search scope to boundaries deduced from hierarchy separation characters specified in the search pattern.

When set to `false`, the commands with `-regexp` parameter will search at the top level only (unless used with the `-hier` parameter).

When set to `true`, the commands will search the specified pattern within the hierarchies based upon the given pattern hierarchy (similar to pattern search without the `-regexp` parameter). The `-hier` along with the `-regexp` parameter shows the same behavior similar to when the command is used without the `-regexp` option - only the leaf level object name is compared to the pattern match.

Examples

Command behavior when this attribute is set to `false`:

```
get_pins -regexp seg2/.*/QN
**WARN: (TCLCMD-513): The software could not find a matching object of the specified type
for the pattern '.*/.*'
**ERROR: (TCLCMD-917): Cannot find 'pins' that match 'seg2/.*.*/QN'

get_pins -regexp seg2/.*/QN -hier
seg2/u3/QN seg2/u9/QN seg2/u14/QN

get_pins -regexp .*/QN -hier
seg1/u3/QN seg1/u9/QN seg1/u14/QN seg2/u3/QN seg2/u9/QN seg2/u14/QN seg3/u3/QN seg3/u9/QN
seg3/u14/QN u2/QN u5/QN u8/QN
```

Command behavior when this attribute is set to `true`:

```
get_pins -regexp seg2/.*/QN
seg2/u3/QN seg2/u9/QN seg2/u14/QN

get_pins -regexp seg2/.*/QN -hier
**WARN: (TCLCMD-513): The software could not find a matching object of the specified type
for the pattern '.*/.*'
**ERROR: (TCLCMD-917): Cannot find 'pins' that match 'seg2/.*.*.*.*.*/QN'

get_pins -regexp .*/QN -hier
seg1/u3/QN seg1/u9/QN seg1/u14/QN seg2/u3/QN seg2/u9/QN seg2/u14/QN seg3/u3/QN seg3/u9/QN
seg3/u14/QN u2/QN u5/QN u8/QN
```

timing_enable_hierarchical_get_nets_support

`timing_enable_hierarchical_get_nets_support {true | false}`

Default: `false`

Enables the hierarchical segment support for nets that are specified using the `get_nets` command.

When set to `false`, the `get_nets` command will report only flat nets.

timing_enable_latch_thru_mode

`timing_enable_latch_thru_mode {true | false}`

Default: `false`

Enables the latch through analysis mode in which timing paths can propagate across latches, depending on their arrival time with respect to the latch transparency window.

When set to `true`, the arrival and required times will be propagated across a latch if it is in transparent mode. The reporting will show the entire flow through the path, rather than individual segments. This allows the overall path slack to be seen through upstream borrowing latch stages. The timing analysis, timing optimization, and reporting will consider the sequence of borrowing latch stages as a single path. As a result, the timing optimization reaches the target slack by viewing the entire path.

This feature provides better visibility into the overall context of latch path timing issues - for both the stage and path-level slacks. Additionally, this improves usability for debugging and manual ECOs, and shows better quality of result for Tempus ECO timing closure.

timing_enable_latency_through_clock_gating

`timing_enable_latency_through_clock_gating {true | false}`

Default: `true`

Allows propagation of latency phases across clock gating elements.

When set to `true`, the software allows propagation of clock latency phase across clock gating elements.

When set to `false`, the software suppresses propagation of clock latency across clock gating elements from the enable pin. The enable pin of the clock gating element should not be in the clock network for any other clocks. This is not recommended because the enable path (through the 'inferred' clock gates, such as, AND, NAND, OR) is either for clock latency or data paths. If the gated clock check is

met, then the enable should not pass through to the output.

In case of ICGs, there is no arc from the EN to the CLKOUT on ICGs that have an incoming enable signal latched. In case of ICG/AND, using the enable path may not be correct for determining latency paths. If no latency paths are found, then the generated clocks definition will need to be re-checked.

timing_enable_mmmc_loop_breaking

```
timing_enable_mmmc_loop_breaking {true | false}
```

Default: `true`

When set to `true`, loop breaking is handled independently per analysis view, according to the following rules in order of precedence:

- If an analysis view does not contain a combinational loop, it does not inherit any loop breaking done for other views.
- If a loop exists, and the user has supplied specific case analysis and `set_disable_timing` constraints to break the loop at a certain place, the specified loop break is honored.
- If a loop is still detected, the software automatically selects a point to break the loop, and applies the loop break consistently to any views where the loop exists.

When set to `false`, if a combinational loop is detected in any analysis view, the timing system breaks the loop by disabling an arc along the looping path. These combinational loop breaks are reflected in all analysis views, even if those views by themselves do not contain these looping paths.

> Cadence strongly recommends that all combinational loops detected by the software be evaluated, to ensure that the loop is broken at the most ideal location. Automatic loop breaking decisions are made based on how the design is traversed. Small changes in the design can influence where loop breaking decisions are made.

To reduce risk and the need to re-evaluate loop-breaking decisions, it is highly recommended that once combinational loop breaks have been evaluated, the user add constraints to ensure that the loop is always broken in that location.

timing_enable_multi_drive_net_reduction_with_assertions

```
timing_enable_multi_drive_net_reduction_with_assertions {none | delay | all}
```

Default: `none`

Allows reduction of multi-drive networks when constraint assertions are present on the network elements:

The following options are available:

- `none`: Reduction is not performed on portions of the network where constraints have been asserted.
- `delay`: Reduction of the multi-drive network will still be applied if only `set_annotated_delay` and `set_annotated_transition` constraints have been applied. Other constraints present on the multi-drive network can still prevent full reduction.
- `all`: The software ignores the presence of any assertions on the multi-drive network and forces reduction to occur.

timing_enable_multi_frequency_latch_analysis

```
timing_enable_multi_frequency_latch_analysis {true | false}
```

Default: `false`

Enables multi-frequency latch timing analysis of the latch time borrowing for when a data signal coming to a latch is controlled by a clock with a frequency different to a clock of the latch enabling signal.

timing_enable_multicycle_data_check_compatibility

```
timing_enable_multicycle_data_check_compatibility {true | false}
```

Default: `false`

Enables checking for SDC compatible data checks when `set_multicycle_path -start` parameter is specified.

timing_enable_path_delay_to_unconstrained_endpoints_compatibility

```
timing_enable_path_delay_to_unconstrained_endpoints_compatibility {true | false}
```

Default: `false`

When set to `true`, the software does not constrain paths with path delay - if the path ends at an unconnected input pin of a combination cell.

timing_enable_pessimistic_cppr_for_reconvergent_clock_paths

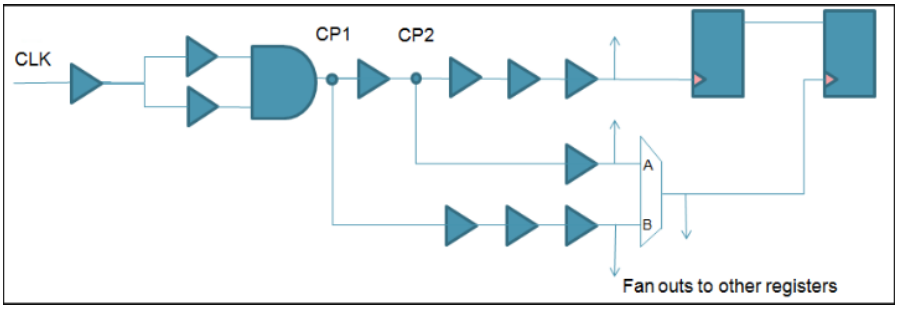
`timing_enable_pessimistic_cppr_for_reconvergent_clock_paths{true | false}`

Default: `false`

When set to `true`, the software will consider all combinations of launch and capture clock path pairs to identify all possible common pins, and then selects the pin with minimum CPPR adjustment over all such common pins as the common pin.

When set to `false`, the default, the software will choose the CPPR common pin based on the commonality of the worst launch and worst capture path only.

Consider the following example,



In the above diagram, both capture paths are considered along with the single launch path, and the common pin is chosen as the pin with the least CPPR adjustment. The pin CP1 is chosen as a common pin and CPPR is reported up to this pin. The same common pin, CP1, will be used for CPPR adjustment whether the capture clock path goes through the mux A pin or the mux B pin. When this attribute is set to false, CPPR common pin is chosen by the commonality in the worst-case launch and worst-case capture paths. so CP2 is chosen as the common pin and CPPR is reported up to this pin.

timing_enable_power_ground_constants

`timing_enable_power_ground_constants {true | false}`

Default: `false`

Determines if case analysis is inferred from power and ground rail connections/nets, e.g., VDD and GND.

When set to `true`, the software recognizes the constants that come from power and ground rail

connections.

When set to `false`, the software ignores these constants. This provides the most intuitive behavior with logical style netlist (and other tools that do not read physical data).

timing_enable_preset_clear_arcs

```
timing_enable_preset_clear_arcs {true | false}
```

Default: `false`

Determines whether the timing arcs are created to model the transition to active state (assertion) of the preset or clear pin, and the subsequent transition of the output to a controlled state.

When set to `true`, the software builds the arcs and the timer automatically analyzes paths through them (enable clock propagation through preset clear pins).

When set to `false`, the software does not build these arcs, because they are associated with analyzing the reset mode timing paths. Filtering them prevents unwanted interaction with functional mode timing analysis.

Note: The `timing_enable_preset_clear_arcs` attribute can also impact delay calculation results of a design. When an arc is disabled, the software will calculate delay using the linear table lookup. When an arc is not disabled, the delay calculations are made based on the quadratic table lookup.

Note: This attribute should be set to `true` when reset checks are performed on a dedicated reset SDC.

timing_enable_pulse_latch

```
timing_enable_pulse_latch {true | false}
```

Default: `true`

Enables pulse-latch analysis.

When set to `true`, the pulse-latch analysis mode provides several additional pulse-clock specific constraint specifications and reporting capabilities.

When in pulse-latch mode, you can use the following pulse clock constraints:

- `set_pulse_clock_max_width`
- `set_pulse_clock_min_width`

- set_pulse_clock_max_transition
- set_pulse_clock_min_transition

Additionally, in this mode, the report_constraint command supports the following pulse clock check types:

- pulse_clock_max_width
- pulse_clock_min_width
- pulse_clock_max_transition
- pulse_clock_min_transition

timing_enable_si_cppr

```
timing_enable_si_cppr {true | false}
```

Default: `true`

Enables more accurate CPPR analysis when incremental delays are present.

When set to `true`, the incremental delay is not included in the CPPR calculation during setup analysis, but is included in the CPPR calculation during hold analysis.

When set to `false`, the incremental delay is included in both setup and hold CPPR calculations.

timing_enable_simultaneous_setup_hold_mode

```
timing_enable_simultaneous_setup_hold_mode {true | false}
```

Default: `false`

Controls whether setup and hold checks are analyzed separately, or together on the same timing graph.

When set to `false`, the software switches the system from analyzing setup slacks to analyzing hold slacks.

When set to `true`, the setup and hold slacks are analyzed concurrently. As a consequence, both the `-early` and `-late` queries for reporting commands, such as report_timing, can be satisfied immediately, without additional calculation. The simultaneous mode is useful for immediately understanding the consequences to both setup and hold slacks from what-if operations and manual ECOs to the design.

Note: The simultaneous setup and hold mode can be used only for timing analysis; it cannot be used

for any physical implementation steps.

Note: This attribute must always be set to `true` while running SI analysis.

timing_enable_timing_window_pessimism_removal

`timing_enable_timing_window_pessimism_removal{true | false}`

Default: `false`

Removes pessimism in SI impact calculation. As a result of elongated arrival windows of attackers and victims, there may be pessimism in SI impact calculation due to a difference in the minimum-maximum arrivals of common clock paths shared between attackers and victims. This pessimism can be removed by adjusting attacker/victim arrival window with the difference in minimum-maximum delay of common clock path between victim and attacker.

When set to `true` (when CPPR is enabled), the software adjusts the timing windows of attackers/victims when the victim is located either on a data path or a clock path during path-based analysis.

For adjustment on victim nets lying on clock paths, full path path-based analysis can be enabled by setting the `timing_disable_retime_clock_path_slew_propagation` attribute to `false`.

timing_enable_tristate_clock_gating

`timing_enable_tristate_clock_gating {true | false}`

Default: `false`

The timing system infers gated-clock checks in the design where a data and clock path intersect at a combinational device. When this attribute is set to `true`, inferred gated-clock checks are added when clock and data signals converge through the tristate enable and data input of tristate buffers. By default (a setting of `false`), gated-clock checks are not added for these situations.

timing_enable_uncertainty_for_clock_checks

`timing_enable_uncertainty_for_clock_checks {true | false}`

Default: `true`

Considers clock uncertainty when performing clock checks.

When set to `true`, the `set_clock_uncertainty` command can supports clock uncertainty for period, pulse width, skew, and separation checks.

timing_enable_uncertainty_for_pulse_width_checks

`timing_enable_uncertainty_for_pulse_width_checks {true | false}`

Default: `false`

When set to `true`, considers clock uncertainty when performing minimum pulse width checks. Clock uncertainty is considered during checks of both user-defined minimum pulse widths (using the `.lib`).

Note: Specifying a value of `true` affects the following commands that take minimum pulse width checks into account:

- `report_timing -check_type pulse_width`
- `report_timing -check_clocks`
- `report_min_pulse_width`
- `report_analysis_coverage`

timing_extract_model_disable_cycle_adjustment

`timing_extract_model_disable_cycle_adjustment {true | false}`

Default: `false`

Controls cycle adjustment for multi-cycle paths.

When set to `false`, if the multi-cycle paths specified in the SDC at the block-level cannot be pushed out to the ports (since there are other paths going through these paths), then the Tempus software adjusts the delay/check numbers in the ETM to take into account the cycle adjustment.

When set to `true`, the cycle adjustment for the multi-cycle paths in ETM arc delay are disabled.

timing_extract_model_aocv_mode

`timing_extract_model_aocv_mode {graph_based | path_based | none}`

Default: none

Allows you to specify AOCV derating mode.

The valid values are described below:

- `graph_based`: Specifies that the delays in the timing model are derated using the graph-based stage counts.
- `path_based`: Specifies that the delays in the timing model are derated using the total path

stage count of worst path between those pin pairs.

- `none`: Specifies that the delays in the timing model are under-rated base delays.

Note: Before setting this attribute to `graph_based or path_based` mode, it is mandatory that the AOCV libraries are read in and AOCV analysis is enabled.

timing_extract_model_case_analysis_in_library

```
timing_extract_model_case_analysis_in_library {true | false}
```

Default: `true`

Enables the software to model the constants as a function in the extracted timing model.

When set to `true`, the software models constants as functions in the library if these propagate to the output port.

When set to `false`, instead of modeling constants as functions inside the library, these are written as `set_case_analysis` in the constraints file (*.asrt).

timing_extract_model_check_arcs_as_lvf

```
timing_extract_model_check_arcs_as_lvf {true | false}
```

Default: `false`

Enables check arcs to be modeled as LVF arcs. By default, the check arcs will be written as nSigma numbers.

The software allows you to write mean+/-nSigma values for constraint arcs, while writing separate mean and sigma values for delay and transition data. This can be enabled by setting `timing_extract_model_check_arc_as_lvf` attribute to `false` and `timing_extract_model_write_lvf` attribute to `true`.

timing_extract_model_consider_design_level_drv

```
timing_extract_model_consider_design_level_drv{true | false}
```

Default : `true`

Considers `set_max_capacitance` and `set_max_transition`DRV constraints, that have been applied on the current design during model extraction.

When set to `true`, these constraints applied at the design level will be considered during model extraction. However, the maximum capacitance and maximum transition from various DRV constraints

applied/inferred on the ports is considered.

When set to `false`, the software extracts the worse-case DRV values and the constraints applied on the current design will be ignored.

timing_extract_model_exhaustive_validation_dir

```
timing_extract_model_exhaustive_validation_dir {true | false}
```

Default: `false`

Controls the location of files to be stored in the ETM Extremity Validation mode.

When set to `false`, additional files generated during ETM Extremity Validation (but not generated during normal validation) will be stored in the current working directory.

When set to `true`, the location to place these files can be user-specified.

timing_extract_model_exhaustive_validation_mode

```
timing_extract_model_exhaustive_validation_mode{true | false}
```

Default: `false`

Enables extracted timing model (ETM) validation at the minimum and maximum indices of the slew load indices.

When set to `false`, the ETM validation is done only for the input slews/output-loads that have been asserted on the design during timing model extraction.

When set to `true`, the software uses the “Extremity-Validation” (EV-ETM) methodology that allows ETM to model the timing arcs in the design for a range of discrete input slews/output-loads, thus providing a more comprehensive validation.

timing_extract_model_gating_as_nochange_arc

```
timing_extract_model_gating_as_nochange_arc {true | false}
```

Default: `true`

Disables conversion of clock gating checks to nochange arcs.

When set to true, the software disables conversion of clock gating checks to nochange arcs in the extracted timing model.

timing_extract_model_ideal_clock_latency_arc

`timing_extract_model_ideal_clock_latency_arc{true | false}`

Default: `false`

When set to `true`, the latency arcs for generated clocks that are defined ideal are extracted with reference as master clock, during ETM extraction.

When set to `false`, arcs are not extracted for generated clocks that are defined ideal during ETM extractions.

timing_extract_model_include_applied_load_in_characterization_range

`timing_extract_model_include_applied_load_in_characterization_range {true | false}`

Default: `true`

Specifies that the actual load visible to the timer that will be included in the load characterization range for path end points.

timing_extract_model_include_applied_slew_in_characterization_range

`timing_extract_model_include_applied_slew_in_characterization_range {true | false}`

Default: `false`

Specifies that the actual timer slew must be included in the characterization range timing model.

timing_extract_model_non_borrowing_latch_path_as_setup

`timing_extract_model_non_borrowing_latch_path_as_setup {true | false}`

Default: `false`

Models the non-borrowing interface latch paths as setup checks.

timing_extract_model_slew_propagation_mode

`timing_extract_model_slew_propagation_mode {worst_slew | path_based_slew}`

Default: `worst_slew`

Specifies the type of slew propagation to use for generating extracted timing model (ETM).

- `worst_slew`: Propagates the worst slew of all the incoming arcs at a converging point for extracting the arcs.
- `path_based_slew`: Propagates the actual slew for the path elements for extracting the arcs.

timing_extract_model_write_clock_checks_as_arc

```
timing_extract_model_write_clock_checks_as_arc{true | false}
```

Default : `false`

When set to `true`, the clock checks (min_period and min_pulse_width) are modelled as arcs in the extracted timing models (ETM). Clock check arcs will be modelled as a single dimensional table with constrained and reference pin as clock pin itself.

When set to `false`, clock checks are modelled as scalar numbers in the extracted ETM.

timing_extract_model_write_clock_checks_as_scalar_tables

```
timing_extract_model_write_clock_checks_as_scalar_tables {true | false}
```

Default: `false`

Allows the extracted timing model to write the minimum pulse width or minimum period checks as scalar arcs.

timing_extract_model_write_lvf

```
timing_extract_model_write_lvf {true | false}
```

Default: `false`

Generates extracted timing model (ETM) containing Liberty Variation Format (LVF) model.

When set to `true`, this attribute enables writing separate mean and variation tables for all the delay/constraint and transition tables.

When set to `false`, the variation data will be merged with the mean values. The mean +/- nsigma values of delay/constraint/transition data will be written in the ETM.

timing_extract_model_write_min_max_clock_tree_path

```
timing_extract_model_write_min_max_clock_tree_path {true | false}
```

Default: `true`

Writes the minimum/maximum worst latency for each clock source.

timing_generated_clocks_inherit_ideal_latency

`timing_generated_clocks_inherit_ideal_latency {true | false}`

Default: `false`

When set to `true`, the software renders generated clocks to inherit parent ideal network latency.

timing_generated_clocks_allow_nested_assertions

`timing_generated_clocks_allow_nested_assertions {true | false}`

Default: `false`

Controls clock latency handling for nested generated clocks that are created on the same timing pin as the master clock.

When set to `true`, the software handles generation of multiple nested generated clocks that are asserted on the same pin, such that the child clock inherits the clock latency value of its parent clock.

When set to `false`, the nested generated clocks asserted on the same pin do not inherit the latency values and are created with 0ns master clock latency. Any full clock timing path reported using these clocks will show the clock path up to the target pin of the generated clock.

timing_generate_normalized_driver_waveform

`timing_generate_normalized_driver_waveform{0 | 1}`

Default: 1

Generates the analytical normalized driver waveform for timing library if the waveform is not specified in the library. The waveform is generated by weighed mean of linear and exponential waveform.

timing_get_of_objects_hier_compatibility

`timing_get_of_objects_hier_compatibility {true | false}`

Default: `false`

When set to `true`, the `get_pins -of_objects` parameter will return leaf pins connected to the specified

objects.

When set to `false`, the `get_pins -of_objects` parameter will return hierarchical pins and leaf pins connected to the specified objects. In order to return leaf pins only when the attribute is set to `false`, the `-leaf` option must be specified along with the `get_pins -of_objects` parameter.

When set to `true`, the `get_cells -objects` parameter returns cells connected to the net at the same level of hierarchy as the net itself. Hierarchical ports will be returned if the net is connected to a hierarchical module.

timing_hier_obj_name_compatibility

```
timing_hier_obj_name_compatibility {true | false}
```

Default: `true`

Controls how hierarchical delimiters are interpreted in the search pattern, when the `-hier` parameter is used with the `get_*`collection commands.

When set to `true`, the software interprets the hierarchical delimiter as part of the search pattern itself, and tries to match names that include the delimiter. For example, if you specify:

```
get_cells -hier L1/R*
```

The software searches at every level of the hierarchy for instances with the name `L1/R*`, completely ignoring the hierarchy separator:

When set to `false`, hierarchical delimiters are interpreted as hierarchy separators. For example, if you specify:

```
get_cells -hier L1/R*
```

The software searched for instances with names matching `R*` in the hierarchy level below `L1`:

```
TL2/L1/RX1 TL2/L1/RX2
```

timing_ignore_lumped_rc_assertions

```
timing_ignore_lumped_rc_assertions {true | false}
```

Default: `false`

When set to `false`, the `set_load` and `set_resistance` values override the actual extracted representation. A warning message is issued stating that the specified constraints on internal nets will override extracted parasitics on those nets.

When set to `true`, instructs the software to ignore `set_load` and`set_resistance` assertions on internal nets in the design. Parasitic data for internal nets normally should be provided by RC extraction, or external SPEF annotation.

Using `set_load` to specify I/O boundary loading on ports is supported by either setting of this attribute.

timing_io_use_clock_network_latency

```
timing_io_use_clock_network_latency {always | ideal}
```

Default: `ideal`

Controls whether network latency of a reference clock is added or not to the data arrival time on the port. When set to `ideal`, the `set_input_delay`and the `set_output_delay`constraints will not add the network latency of the reference clock to the data arrival time on the port if the clock is in `propagated` mode.

When this attribute is set to `always`, network latency is added to the data arrival time on the port regardless of the clock propagation mode.

timing_library_build_async_deassert_arc

```
timing_library_build_async_deassert_arc {true | false}
```

Default: `true`

Controls whether input to output arcs from the pre-set or clear pins transitioning to an active state are included, when the timing system is initialized.

timing_library_convert_async_setuphold_to_recrem

```
timing_library_convert_async_setuphold_to_recrem {true | false}
```

Default: 1

Controls whether single-edged setup and hold checks on asynchronous pins in the Liberty library are inferred as recovery and removal checks.
This attribute provides support for older modeling styles in legacy libraries.

timing_library_generated_clock_use_group_name

```
timing_library_generated_clock_use_group_name {true | false}
```

Default: `false`

When set to `true`, the software uses the generated_clock group name when creating a generated clock from a library-generated clock group. When set to `false`, the software uses the pin name when creating the generated clock.

For example, assume the cell for instance `a/b` in the timing library contained the following generated clock group:

```
generated_clock(genclk2) {
clock_pin : clk1;
master_pin : "int/clk";
multiplied_by : 2;
duty_cycle : 50.0;
}
```

By default (`false`), the software creates a generated clock with the following name:

```
a/b/clk1
```

If the `timing_prefix_module_name_with_library_generated_clock` attribute is set to `true`, and you set this attribute to `true`, the software creates a generated clock with the following name:

```
a/b/genclk2
```

If the `timing_prefix_module_name_with_library_generated_clock` attribute is set to `false`, and you set this attribute to `true`, the software creates a generated clock with the following name:

```
genclk2
```

timing_library_hold_constraint_corner_sigma_multiplier

`timing_library_hold_constraint_corner_sigma_multiplier` *`double`*

Default: 0

Applies user control multiplier to generate a constraint table using sigma values applied for hold arc.

timing_library_hold_sigma_multiplier

`timing_library_hold_sigma_multiplier` *`double`*

Default: 0.0

User control multiplier to generate constraint table applied for hold arc.

timing_library_infer_async_pins_from_timing_arcs

```
timing_library_infer_async_pins_from_timing_arcs {true | false}
```

Default: *false*

When set to `true`, a pin is marked clear if it has timing arcs with type “clear” to output pins. The sequential block is not required to be present in the cell for marking a pin as clear.

timing_library_infer_cap_range_from_ccs_receiver_model

```
timing_library_infer_cap_range_from_ccs_receiver_model {true | false}
```

Default: `false`

Infers the rise or fall of capacitance range values from CCS receiver capacitance model.

When set to `false`, the rise/fall CCS capacitance range values are not inferred. The NLDM pin capacitance range values defined in the library are honored.

timing_library_infer_cap_range_from_ecsm_receiver_model

```
timing_library_infer_cap_range_from_ecsm_receiver_model {true | false}
```

Default: `false`

Infers the rise or fall of NLDM pin capacitance range from the ECSM capacitance range while calculating the total loading capacitance during switching power computation. If the attribute is set to `true` and the capacitance range is not defined in the library, the rise/fall capacitance range is inferred from the ECSM receiver capacitance models. The capacitance range values are derived either from the pin-level ECSM cap models or the timing-level ECSM cap models.

When this attribute is set to `false`, the rise/fall capacitance range values are not inferred, and the library values are honored.

timing_library_interpolate_drv_values

```
timing_library_interpolate_drv_values  {true | false}
```

Default: `false`

Allows the software to use a range of trilib DRV values for performing delay calculations.

timing_library_infer_socv_from_aocv

```
timing_library_infer_socv_from_aocv {true | false}
```

Default: `false`

Controls the transformation of AOCV data.

This attribute infers sensitivity data from AOCV libraries, and must be set before loading any timing library data.

timing_library_read_ccs_noise_data

```
timing_library_read_ccs_noise_data{true | false}
```

Default: `true`
This attribute can be used when the CCS library contains noise models, and SI glitch analysis is being performed.

timing_library_read_without_ecsm

```
timing_library_read_without_ecsm {true | false}
```

Default: `false`

Disables loading ECSM data.

When set to `true`, the software allows you to change the mode of use (of ECSM timing model) if the timing library contains NLDM, ECSM, or CCS data in the same file.

Note: This attribute does not disable CCS data loading.

timing_library_read_without_power

```
timing_library_read_without_power {true | false}
```

Default: `false`

When set to `true`, this attribute allows the software to read timing libraries without power information. When set to `false`, the software reads power data also.

timing_library_read_without_sensitivity

```
timing_library_read_without_sensitivity {true  | false}
```

Default: `false`

Disables loading of ECSM timing sensitivity data.

timing_library_scale_aocv_to_socv_to_n_sigma

`timing_library_scale_aocv_to_socv_to_n_sigma<`*`n_sigma value`*`>`

Default: 3.0

AOCV derates are expected to be derived based on 3-sigma variation.

timing_library_setup_constraint_corner_sigma_multiplier

`timing_library_setup_constraint_corner_sigma_multiplier` *`double`*

Default: 0

Applies user control multiplier to generate a constraint table using sigma values applied for setup arc.

timing_library_setup_sigma_multiplier

`timing_library_setup_sigma_multiplier` *`double`*

Default: 0.0

User control multiplier to generate constraint table using sigma values applied for setup arc.

timing_library_term_voltage_from_lib_pin

`timing_library_term_voltage_from_lib_pin` *`int`*

Default: `0`

Enables the use of library pin voltage for reporting the term voltage.

timing_library_use_two_piece_receiver_cap

`timing_library_use_two_piece_receiver_cap {true | false}`

Default: `false`

Enables the use of 2-piece receiver capacitance constructs, if a library contains both the N-piece and 2-piece receiver capacitance construct.

When set to `true`, if a library has 2-piece and multi-piece pin capacitance, then the software gives priority to 2-piece models.

When set to `false`, the software reads the N-piece receiver capacitance constructs.

timing_library_zero_negative_timing_check_arcs

`timing_library_zero_negative_timing_check_arcs {true | false}`

Default: `false`

When set to `true`, the software checks for negative values in the timing arcs.

timing_multi_frequency_clock_rounding_factor

`timing_multi_frequency_clock_rounding_factor` *`double`*

Default: `1e-05`

Rounds off irrational numbers so that they can be read appropriately.

timing_normalized_driver_waveform_clip_linear_part

`timing_normalized_driver_waveform_clip_linear_part {0 | 1}`

Default: `0`

When set to 1, the linear ramp component in the normalized driver waveform is saturated to the maximum value, while generating the normalized driver waveform.

timing_normalized_driver_waveform_weight_factor

`timing_normalized_driver_waveform_weight_factor` *`double`*

Default: 0,5

Defines the value of the weight of linear components in the algorithm for generation of normalized driver waveform.

When set to `1`, the waveform will be purely linear. When set to `0`, the waveform will be purely exponential. You can specify values within the range of 0 to 1.

timing_nsigma_multiplier

`timing_nsigma_multiplier` *double*

Default: 3.0

Controls the sigma multiplier to be used in SOCV mode.

timing_null_collection_return_compatibility

`timing_null_collection_return_compatibility {true | false}`

Default: `false`

Allows you to migrate previous release scripts to the new use model.

When the attribute is set to `true`, the software returns "NA" values that are consistent with the previous release. The nil return values are returned as an empty collection. This allows the return value to be used directly by collection iterators, such as `foreach_in_collection`.

The default value is `false`, which is consistent with the current release.

Note: This compatibility option may be removed from future releases. You should update your scripts to the new use model to avoid future problems.

timing_path_based_enable_bounding_box_for_io_paths

`timing_path_based_enable_bounding_box_for_io_paths {true | false}`

Default : `false`

Enables the bounding box mode for input-output paths.

timing_path_based_exhaustive_pba_bounded_mode

`timing_path_based_exhaustive_pba_bounded_mode {true | false}`

Default: `false`

Enables the IPBA mode with exhaustive PBA analysis (run using the `report_timing -retime_mode exhaustive` command).

When set to `true`, the software ensures that all the paths to the violating endpoints are covered for PBA, and the slack is the worst PBA slack reported across all the paths to the endpoint. For any endpoint that is considered for IPBA analysis, the complete fanin cone (all the timing paths ending at that endpoint) is evaluated for any possible PBA violation.

timing_path_based_enable_closest_common_pin_driver_for_bounding_box

`timing_path_based_enable_closest_common_pin_driver_for_bounding_box {true | false}`

Default : `false`

Enables bounding box calculation using closest common pin driver from the clock mesh.

timing_path_based_enable_exhaustive_depth_bounded_by_gba

`timing_path_based_enable_exhaustive_depth_bounded_by_gba {true | false}`

Default: `true`

Enables the timing report to bound PBA results by GBA slacks of the respective leftover nworst paths, whenever exhaustive PBA depth for an endpoint is exhausted.

When set to `true`, during exhaustive PBA, if you specify exhaustive depth as N, and an endpoint has nworst paths greater than N, then the timing report will show the worst path corresponding to worst slack out of worst PBA slack (out of first N paths) and GBA slack (for the (N+1)th path). Here, the variable N refers to exhaustive PBA depth and can be set using the `timing_pba_exhaustive_path_nworst_limit` attribute.

When set to `false`, during exhaustive PBA, if you specify exhaustive depth as N, and an endpoint has nworst paths greater than N, then the software will ignore the paths with depth greater than N, which may result in reporting an optimistic PBA slack for the endpoint reaching the depth.

timing_path_based_enable_report_launch_clock_path

`timing_path_based_enable_report_launch_clock_path {true | false}`

Default: `true`

When set to `false`, the software skips the reporting launch clock path in PBA path_type full report.

timing_path_based_enable_verbose_mode

`timing_path_based_enable_verbose_mode` *`value`*

Default: `1`

Controls printing of verbose messages during and at the end of path-based analysis (PBA).

You can specify one of the three values - 0, 1, and 2. These are described below:

- `0`: Verbose messages are not reported during exhaustive PBA analysis. For default PBA analysis, the path count will be reported.
- `1`: Reports the final message for PBA path count and max depth reached during exhaustive PBA. Progress messages are not reported.
- `2`: Reports multiple progress messages after regular intervals (in addition to the final message for PBA path count and max depth).

This attribute impacts messages when the `report_timing` and `report_constraint` commands are used in PBA mode.

timing_path_based_exhaustive_enable_design_coverage

```
timing_path_based_exhaustive_enable_design_coverage
{true | false}
```

Default: `false`

Enables exhaustive path-based analysis (PBA) to automatically pick up and analyze all the violating endpoints in the design, irrespective of the specified maximum paths.

When set to `true`, exhaustive PBA analysis automatically picks up and analyzes the violating design endpoints instead of N endpoints (specified with the `report_timing -max_paths` parameter). After analyzing the violating endpoints, the software reports the top N worst endpoints - depending on the specified value of N.

When set to `false`, exhaustive PBA analysis restricts analysis (using the `-max_paths` parameter) to the specified number of N endpoints.

Note: When this attribute is enabled, there may be an increase in run time because the software will analyze all the violating endpoints and analysis will not be restricted to the specified N endpoints.

timing_path_based_exhaustive_max_paths_limit

```
timing_path_based_exhaustive_max_paths_limit integer
```

Default: `2000000`

Specifies the maximum number of paths that can be internally analyzed during exhaustive path-based

analysis (PBA).

If the number of candidate paths during exhaustive PBA is greater than the specified max paths limit, then the software will skip path-based analysis of these paths beyond this limit. The paths exceeding the specified max paths limit will then be included as GBA in the final report output so that real violations are not missed.

timing_pba_exhaustive_path_nworst_limit

`timing_pba_exhaustive_path_nworst_limit <value>`

Default: `10000`

Specifies the maximum number of paths per endpoint (nworst) that will be re-timed in exhaustive path-based analysis (PBA).

In exhaustive PBA mode, the value specified with the `report_timing -nworst` parameter is used only during the final reporting. The value specified with the `timing_pba_exhaustive_path_nworst_limit` attribute (and not the `report_timing -nworst` value) is used to specify the maximum number of paths to be re-timed per endpoint.

If the limit specified by `timing_pba_exhaustive_path_nworst_limit` is reached and `timing_path_based_enable_exhaustive_depth_bounded_by_gba` attribute is set to:

- `true`: The worst of the current PBA slack is used and N+1th GBA path is reported.
- `false`: The current PBA slack is reported - which may be optimistic.

timing_pll_clock_use_driving_cell

```
timing_pll_clock_use_driving_cell
{true | false}
```

Default : `false`

Uses delay calculation of all the enabled arcs to the PLL clock output to compute the transition.

When set to `true`, the delay calculation of all enabled arcs to the PLL clock output will be used to compute the transition. If no driving arcs are available in the model or if they are disabled, the same idealized transition will be used.

When set to `false`, the software considers the PLL clock output to be electrically isolated from the input-side reference clock, and creates the output clock with an idealized transition. You can use the `set_annotated_transition` command to customize the output clock transition.

timing_prefix_module_name_with_library_generated_clock

`timing_prefix_module_name_with_library_generated_clock{true | false}`

Default: `true`

When set to `true`, the software appends the instance name to the clock pin name when creating a generated clock. When set to `false`, the software uses only the clock pin name when creating a generated clock.

For example, assume the cell for instance `a/b` in the timing library contained the following generated clock group:

```
generated_clock(genclk2) {
clock_pin : clk1;
master_pin : "int/clk";
multiplied_by : 2;
duty_cycle : 50.0;
}
```

By default (`true`), the software creates a generated clock with the following name:

```
a/b/int/clk
```

If you set the attribute to `false`, the software creates a generated clock with the following name:

```
int/clk
```

If you set this attribute to `false`, and the `timing_library_generated_clock_use_group_name` attribute is set to `true`, the software creates a generated clock with the following name:

```
genclk2
```

timing_propagate_latch_data_uncertainty

`timing_propagate_latch_data_uncertainty {true | false}`

Default: `false`

When set to `true`, uses the clock phase associated with a flush latch's data pin as the "from" clock phase for downstream uncertainty timing calculations

When set to `false`, the downstream uncertainty calculations are based on the clock phase of the flush latch's clock signal.

timing_property_arrival_clocks_consider_clock_source_data_phase

`timing_property_arrival_clocks_consider_clock_source_data_phase {true | false}`

Default : `true`

Considers the clock source data phases.

The `get_property` and `report_property` commands support the `arrival_clocks` collection property on pin objects. The `arrival_clocks` property considers data phases, clock source data phases, and latency phases.

timing_property_arrival_clocks_consider_data_phase

`timing_property_arrival_clocks_consider_data_phase {true | false}`

Default: `true`

Returns the pure data phase.

The `get_property` and `report_property` commands support the `arrival_clocks` collection property on pin objects. The `arrival_clocks` property considers data phases, clock source data phases, and latency phases.

timing_property_arrival_clocks_consider_latency_phase

`timing_property_arrival_clocks_consider_latency_phase {true | false}`

Default: `true`

Returns the latency phase.

The `get_property` and `report_property` commands support the `arrival_clocks` collection property on pin objects. The `arrival_clocks` property considers data phases, clock source data phases, and latency phases.

timing_property_arrival_window_enable_tcl_dict_format

`timing_property_arrival_window_enable_tcl_dict_format {true | false}`

Default : `false`

Allows the software to print the new format of the `arrival_window` property

(see `get_property` command).

timing_property_return_null_collection_with_quiet

`timing_property_return_null_collection_with_quiet {true | false}`

Default: `false`

Allows you to query values of all the properties.

When set to `true`, the software enables returning values of properties of all the objects, when the `get_property -quiet` parameter is specified.

timing_rail_swing_checks_high_voltage_threshold

`timing_rail_swing_checks_high_voltage_threshold d`*`ouble`*

Default: `0.95`

Specifies the voltage threshold values for high rail swing checks.

timing_rail_swing_checks_low_voltage_threshold

`timing_rail_swing_checks_low_voltage_threshold d`*`ouble`*

Default: `0.05`

Specifies the voltage threshold values for low rail swing checks.

timing_recompute_sdf_in_setuphold_mode

`timing_recompute_sdf_in_setuphold_mode {true | false}`

Default: `false`

Controls the recomputing of SDF delays when the software is in simultaneous setup and hold analysis mode.

When the software is not in simultaneous setup and hold analysis mode, only one corner of data is stored at a time; therefore additional delay calculation is required for `write_sdf` command. When the software is in simultaneous setup and hold analysis mode, both sets of data are in memory at the same time, and additional delay calculation is not necessary.

If set to `false`, `write_sdf` will write out the delays from memory, instead of calling delay calculation.

Note: This attribute must be used with `timing_enable_simultaneous_setup_hold_mode true`.

timing_reduce_multi_drive_net_arcs

`timing_reduce_multi_drive_net_arcs {true | false}`

Default: `true`

Controls whether multi-drive or mesh networks are reduced to simpler, single driver models.

When set to `true`, the nets that meet the drivers' receivers arc threshold (defined using `timing_reduce_multi_drive_net_arcs_threshold` attribute), are reduced. The net arcs are created only from one driver or from the drivers for which you have set the constraints.

If you specify constraints on the net driven by parallel buffers, it might result in increased run time and memory degradation. By specifying timing constraints through nets, the constraints are moved to the drivers. This leads to the creation of net arcs from all the drivers, thus disabling net arc reduction. In such cases, you can select one driver of the net and use that driver to specify the constraints.

When set to `false`, the number of net arcs created for timing analysis for nets driven by parallel buffers are not reduced. If a net is driven by `n` parallel buffers and is connected to `m` sink pins, then the total number of net arcs created is `n*m`. For example, if `n` is `100` and `m` is `20000`, the number of net arcs created for timing analysis will be two million. This may result in increased run time and memory usage.

timing_reduce_multi_drive_net_arcs_threshold

`timing_reduce_multi_drive_net_arcs_threshold` *`int`*

Default: `10000`

Sets a threshold number that is used to trigger the reduction of timing arcs of nets driven by parallel buffers. The reduction takes place if the number of arcs is greater than the threshold value and if the `timing_reduce_multi_drive_net_arcs` attribute is set to `true`.

timing_report_analysis_summary_csv_extended_new_format

`timing_report_analysis_summary_csv_extended_new_format {true | false}`

Default: `false`

Keeps the analysis summary CSV extended report in sync with the summary timing report. These reports are available when the respective `report_timing` parameters are specified:

- `-analysis_summary_csv_extended_file`

- `-analysis_summary_file`

When set to `true` (and when the `report_timing -analysis_summary_csv_extended_file` parameter is specified), the column name sequence will be from fromClk to toClk, that is, fromClk will be printed first, and then toClk. Also, the timing report will show "-", if toClk or fromClk is not present (and when `report_analysis_summary -expanded_summary_clocks` is set to `false`).

When set to `false`, the missing toClk or fromClk column will be represented as “*”.

timing_report_check_timing_unconstrained_endpoints_due_to_constants

`timing_report_check_timing_unconstrained_endpoints_due_to_constants {true | false}`

Default: `false`

Enables the software to flag warnings for unconstrained signals arriving at the end points due to case analysis.

When set to `false`, the `check_timing` command output will show warnings for the unconstrained endpoints.

When set to `true`, the command output will also show warnings for unconstrained endpoints due to case analysis under the subcategory `uncons_endpoints_due_to_case_analysis`. For more details, see `check_timing`.

timing_report_clock_pin_as_begin_point

`timing_report_clock_pin_as_begin_point {true | false}`

Default: `false`

Allows the software to report clock pins as begin points.

timing_report_constraint_enable_extended_drv_format

`timing_report_constraint_enable_extended_drv_format {true | false}`

Default: `false`

Enables reporting of `report_constraint` as per clock/data phases and rise/fall.

timing_report_constraint_rise_fall_clock_period_check

`timing_report_constraint_rise_fall_clock_period_check {true | false}`

Default: `false`

Enables reporting of rise-to-rise and fall-to-fall clock period checks separately for the `report_constraint` command.

When set to `false`, the software reports the rise-to-rise and fall-to-fall clock period checks under the same table in the report. With the `false` setting, when the `report_constraint -check_type clock_period` option is specified, the software reports violations that are the worst amongst all the rise-fall transitions for each point.

timing_report_disable_max_paths_per_group

`timing_report_disable_max_paths_per_group {true | false}`

Default: `false`

Controls the interpretation of the `report_timing -max_paths` parameter when you request reporting across a set of path groups. By default (`false` setting), the specified `-max_paths` will be reported per path group. With the `true` setting, a total of the specified `-max_paths` will be reported across all of the specified path groups.

For Stylus flow cross-tool compatibility – you should specify a setting of `true`. For backward compatibility with the legacy UI flow – you should use a value of `false`.

For example,

When set to `false`:

Legacy UI: `report_timing -path_group {reg2reg in2reg} -nworst 2 ...`
Stylus UI: `report_timing -group {reg2reg in2reg} -nworst 2 ...`

The `report_timing` command will return up to 2 paths for the reg2reg group and also up to 2 paths for the in2reg group.

When set to `true`:

Legacy UI: `set timing_report_disable_max_paths_per_group true`
Stylus UI: `set_db timing_report_disable_max_paths_per_group true`

A total of 2 paths will be reported across both path groups. The paths may come entirely from one or the other group – or both, based on the slack value.

timing_report_drv_enable_clock_source_as_clock

`timing_report_drv_enable_clock_source_as_clock {true | false}`

Default: `false`

Enables the clock DRV for pure clock source path instead of data DRV.

timing_report_drv_enable_frequency_per_view

`timing_report_drv_enable_frequency_per_view {true | false}`

Default: `false`

Enables view-based frequency DRV flow.

When set to `true`, in the MMMC DRV flow, the software performs frequency computation for design pins as per the clock period of specific analysis views reaching a pin. In case of non-activity based flow, the frequency is computed as per the fastest clock of all the views.

When set to `false`, in the propagate activity flow, the software performs frequency computation for design pins from propagated activity on that pin.

For other flows. the software performs frequency computation for design pins as per the smallest clock period (largest clock frequency) from all the active analysis views.

timing_report_drv_enable_slew_threshold_scaling

`timing_report_drv_enable_slew_threshold_scaling {true | false}`

Default: `false`

Enables the transition constraints scaling as per the slew threshold value.

timing_report_enable_cppr_point

`timing_report_enable_cppr_point { true | false}`

Default: `false`

Enables the timing report to output CPPR point details.

When set to `true`, the `report_timing` command output shows the CPPR point field.

timing_report_enable_flag_field_symbols

```
timing_report_enable_flag_field_symbols  {true | false}
```

Default: `false`

Enables reporting of Instance/Net related bits in the `report_timing` `Flags` column.

timing_report_enable_lead_trail_to_rise_fall_map

```
timing_report_enable_lead_trail_to_rise_fall_map {true | false}
```

Default: `false`

Enables reporting of rise/fall edge times, instead of lead/trail edge times of a clock (reported using the`report_clocks`command).

timing_report_enable_markers

```
timing_report_enable_markers{true | false}
```

Default: true

Enables display of markers in reports.

When set to `true`, the `report_timing`output format displays markers that identify pins specified using the `-from`, `-to`, or `-through` parameters. The markers, displayed as '`->`', show up in the 'timing point' or ' hpin' column (if the 'timing point' column is not available) of the timing report. This feature provides better readability. When set to `false`, the timing report does not display the markers.

timing_report_enable_max_capacitance_drv_for_constant_nets

```
timing_report_enable_max_capacitance_drv_for_constant_nets {true | false}
```

Default: `false`

Enables maximum capacitance DRV checks for nets having disabled constants.

timing_report_enable_max_path_limit_warning

```
timing_report_enable_max_path_limit_warning {true | false}
```

Default: `false`

Determines if the number of reported paths for a particular clock group equals the specified maximum paths limit.

When set to`true`, the software will issue a warning message in case the number of reported paths for a clock group is equal to the given maximum paths limit. This applies to both graph- and path-based reporting modes.

This attribute is supported in clock group-based non-statistical reporting flows only. The clock group-based reporting mode can be enabled by setting `timing_report_group_based_mode` attribute to `true`.

timing_report_enable_report_clock_timing_across_clock_pin

```
timing_report_enable_report_clock_timing_across_clock_pin {true | false}
```

Default: `false`

Enables fanout tracing across clock pins if any combinational arc is present.

timing_report_enable_si_debug

```
timing_report_enable_si_debug {true | false}
```

Default: `false`

Allows the software to capture signal integrity (SI) related details for each stage of a path being analyzed during path-based analysis in the form of attributes.

timing_report_enable_unique_pins_multiple_capture_clock_paths

```
timing_report_enable_unique_pins_multiple_capture_clock_paths {true | false}
```

Default: `false`

Enables reporting of different paths with respect to the reference clocks in the unique pins.

timing_report_enable_verbose_ssta_mode

```
timing_report_enable_verbose_ssta_mode {true | false}
```

Default: `false`

Enables reporting of sigma/multiplier for the timing report (`report_timing`command) header section in SSTA or SOCV mode.

timing_report_fields

`timing_report_fields` *`string`*

Default: `{instance arc cell delay arrival required}`

Specifies a `report_timing` format. See `report_timing -fields` for more information.

Valid format fields are:

adjustment, abs_delay_err, abs_transition_err, annotation, aocv_adj_stages, aocv_derate, aocv_weight, arc, arrival, arrival_mean, arrival_sigma, cell, delay, delay_mean, delay_sigma, direction, edge, fanin, fanout, flags, hpin, incr_delay, instance, instance_location, latch_window, load, module, net, pct_delay_err, pct_slew_err, phase, phys_info, pin, pin_load, pin_location, power_domain, required, retime_delay, retime_delay_mean, retime_delay_sigma, retime_slew, retime_transition_mean, retime_transition_sigma, stage_count, transition, transition_mean, transition_sigma, socv_derate, spice_arrival, spice_delay, spice_slew, stolen, timing_point, total_derate, user_derate, when_cond, wire_load, wlmodel

timing_report_generated_clock_info

`timing_report_generated_clock_info {true | false}`

Default: `true`

When set to `true` (the default), generated clock information is automatically added to the report if generated clocks are encountered in either the launching or latching clock paths. The generated clock information column is displayed if the `Delay' field exists. If the `Delay' field does not exist, then the generated clock adjustment column is added to the report only when explicitly specified using the `report_timing -fields` command or by setting the `timing_report_fields` attribute.

timing_report_group_based_mode

`timing_report_group_based_mode {true | false}`

Default: `false`

Groups paths by clock domain.

When set to `true`, the behavior of the following reporting commands will change as follows:

`report_timing`

The behavior of the following `report_timing`parameters changes as follows:

- `-max_points`: In this mode you are not required to specify this parameter. If specified, the

parameter is mapped to `-max_paths` option.

- `-max_paths M`: Reports `M` paths per clock group.
- `-nworst N`: Limits the number of paths per endpoint per clock group. By default, the value of `-nworst` is `1`.

The `report_timing` command will have the following four types of groups (if design does not have user-defined path groups):

- Asynch group: Recovery/removal checks.
- Clock gating group: Inferred and library clock gating checks.
- Default group: Combinational max delay or min delay paths.
- Clock Groups: Groups for all design clocks. For a design with three clocks, three groups will be reported, one group for each clock. Endpoints sharing the same reference clock will be part of the same group. If an endpoint belongs to multiple groups then it will be reported separately in each group.

Note: If user-defined path groups are specified, the software will override the path groups described above.

`report_min_pulse_width`

The `report_min_pulse_width` command reports pulse width slacks of all the clocks for a pin. In summary mode, the "clock" column will be displayed. In verbose mode (with `path_type` option), the behavior will be similar to the `report_timing` output in group-based mode.

`report_constraint`

The `report_constraint`command will report the following checks:

- For max_delay/setup and min_delay/hold type checks paths will be reported for each clock group separately.
- The behavior of pulse width check types will be the same as `report_min_pulse_width` command.

`report_analysis_coverage`

The `report_analysis_coverage` command will report a check at end points for all the clocks. In verbose mode, the "clock" column will be displayed.

timing_report_max_transition_check_using_nsigma_slew

```
timing_report_max_transition_check_using_nsigma_slew
{true | false}
```

Default: `false`

Enables the `report_constraint` command to use mean and sigma slew values for reporting transition values in DRV checks.

When set to `true`, in the SOCV flow the `report_constraint` command uses the mean plus n-sigma slew values for DRV checks.

When set to `false`, the `report_constraint` command will report only the mean slews.

timing_report_property_fastest_clock_consider_data_phase

```
timing_report_property_fastest_clock_consider_data_phase {true | false}
```

Default: `false`

Considers the data signals while determining the fastest clock arriving on any given pin.

When set to `false`, the clocks arriving in clock phase form will be considered for determining the fastest clock.

When set to `true`, the clocks arriving in clock phase as well as data phase form will be considered.

timing_report_pulse_width_matching_launch_capture_paths

```
timing_report_pulse_width_matching_launch_capture_paths {true | false}
```

Default: `false`

Reports pulse width paths that have matching launch and capture paths - the count of pins and the pins must match exactly. The paths that do not meet this condition will not be reported.

When set to `true`, the software will impact the minimum pulse width report that is generated with the `report_min_pulse_width` command along with the `-path_type short` and `-path_type full_clock` options.

timing_report_redirect_message_types

```
timing_report_redirect_message_types
{none | info}
```

Default: `none`

Redirects reporting messages to a report file.

When set to `info`, the path count and max depth (displayed when the `report_timing` command is run in PBA mode) will be appended to the final report output.

When set to `none`, the reporting messages are not redirected to a report file.

timing_report_retime_formatting_mode

```
timing_report_retime_formatting_mode {manual | retime_compare | retime_replace}
```

Default: `retime_replace`

Manages the re-timing fields in the reporting format. Based on the given settings, this attribute automatically replaces (or adds) the default columns with the respective re-timing columns. For example, in case of delay column, 'Retime Delay' will be added or replaced automatically.

timing_report_skip_constraint_loop_check

```
timing_report_skip_constraint_loop_check {true | false}
```

Default: `false`

Allows you to skip loop checking when the `check_timing` command is used.

When set to `true`, the software skips reporting of loops in case the timing graph inside a loop is broken. This may happen if some of the commands, such as, `create_clock`, `set_input_delay`, or `set_output_delay` are defined inside a loop.

Note: When this attribute is set to `true`, the `check_timing` command will report lesser loops than the ones reported in the CTE_loops.rpt report - which reports all the loops.

timing_report_socv_summary_mean_sigma

```
timing_report_socv_summary_mean_sigma {true | false}
```

Default: `false`

Prints the report header in new format when the `report_timing` command is run in SOCV mode.

timing_report_timing_header_detail_info

`timing_report_timing_header_detail_info {default | extended}`

Default: `default`

Controls the timing report (generated using the `report_timing` command) header format. You can specify one of the following options:

- When set to `default`, the software generates timing reports with a report header that includes the following information:
  - Software version number
  - Platform on which the software is being run
  - Time when the report was generated
  - Command used to generate the report
- When set to `extended`, the software generates timing reports with additional header information given below:
  - The analysis mode - best case worst case or on chip variation
  - The active views (displayed only in MMMC mode)
  - The clock reconvergence pessimism (CPPR)
  - The latch slack mode
  - The specified clock propagation value
  - The specified delay calculation engine

Note: You can use this attribute when `timing_disable_report_header_info` is set to `false`.

timing_report_unconstrained_paths

`timing_report_unconstrained_paths {true | false}`

Default: `false`

When set to `true`, the `report_timing` command reports unconstrained paths if the software cannot find any constrained paths to report.

When set to `false`, the `report_timing` command reports only the constrained paths.

Note: The behavior of `timing_report_unconstrained_paths` attribute is not the same as the behavior of `report_timing -unconstrained` command. If there are several paths to the same endpoint from different begin points, including constrained and unconstrained paths, the `report_timing -unconstrained` parameter always reports unconstrained paths, even if a constrained path can be reported. Whereas the `timing_report_unconstrained_paths` attribute reports constrained paths but not unconstrained paths.

timing_report_unconstrained_path_early_late_header

`timing_report_unconstrained_path_early_late_header {true | false}`

Default: `false`

When set to `true`, this software provides information in the timing report header that indicates whether an unconstrained path is a late or early path.

timing_report_use_receiver_model_capacitance

`timing_report_use_receiver_model_capacitance {true | false}`

Default: `false`

Enables the max_capacitance violations based on dynamic pin capacitance.

When set to `true`, the software checks for both the input level and timing arc level dynamic pin capacitance tables for worst-case pin capacitance table for given input slew. If conditional pin capacitance tables (or timing_arcs) exist, the worst-case is used among all the conditions.

This attribute performs static pin capacitance inference after delay calculation, so the delay computation is not impacted. The pin capacitance is used only for reporting and max_cap violations.

timing_report_use_worst_parallel_cell_arc

`timing_report_use_worst_parallel_cell_arc {true | false}`

Default: `false`

When set to `false`, setting the `-nworst` parameter reports several paths by selecting parallel arcs in the library cell between two pins.

When set to `true`, only a single worst path is reported by selecting the worst delay arc among all parallel arcs of a cell. For example, for reporting timing through A to Y pins (where the cell is a 2-input

cell), the parallel arc is applicable for the same input and output pin. So the worst arc for A->Y is chosen from all arcs of A->Y.

timing_resolve_driver_conflicts

`timing_resolve_driver_conflicts {conservative | aggressive}`

Default: `aggressive`

Resolves constant propagation results on nets with multiple drivers.

When the attribute is set to `conservative` mode, the net values are as follows:

- If all the drivers are 1, the net will be driven to value `1`
- If all the drivers are 0, the net will be driven to value `0`
- In other cases, the net value will be `Undefined`

When the attribute is set to `aggressive` mode, the net values are as follows:

- If some of the drivers are 0, the net value will be `0`
- If some of the drivers are 1, but none are 0, then the net value will be `1`
- In other cases, the net value will be `Undefined`

timing_scaling_for_negative_checks

`timing_scaling_for_negative_checks {default | divider | multiplier}`

Default: Negative check * (2.0 - derating factor), where the derating factor is the value that you set using the `set_timing_derate` command.

Modifies the scaling of the negative timing analysis check value. This variable does not affect the positive checks.

When set to `divider`, divides the negative check value by the derating factor.

When set to `multiplier`, multiplies the negative check value by the derating factor.

timing_scaling_for_negative_delays

`timing_scaling_for_negative_delays`
`{default | divider | multiplier}`

Default: Negative delay * (2.0 - derating factor), where the derating factor is the value that you set using the `set_timing_derate` command.

Modifies the scaling of the negative delay value to be used during timing analysis. This variable does not affect the positive delay.

When set to `divider`, divides the negative delay value by the derating factor.

When set to `multiplier`, multiplies the negative delay value by the derating factor.

Currently, constraints are used during timing in the flattened mode, so the internal ILM instances are seen instead of the LEF pins of the ILMs. Therefore, reading the bounding box constraints causes errors without this variable set. Notice that if you want to see the LEF pins of the ILM in GUI, the design must be in the unflattened mode.

timing_sdf_adjust_negative_setuphold

```
timing_sdf_adjust_negative_setuphold {true | false}
```

Default: `false`

Controls how the setup and hold check values are adjusted when both are initially negative in the SDF file.

When set to `true`, the software sets the smaller check delay value to a negated larger check delay value. For example, if the initial setup/hold value is equal to -1/-2. then the value -2 from the setup is smaller and it is reset to a negated value of the larger one (-1 from hold). So the adjusted setup/hold value is equal to -1/1.

When set to `false`, both the negative setup and hold check values are set to 0.

The command order for using this attribute is as follows:

- `read_sdf`
- `timing_sdf_adjust_negative_setuphold {true | false}`
- `write_sdf`

The timing report (`report_timing`) results are not impacted.

timing_sdf_enable_setuphold_scond_ccond

```
timing_sdf_enable_setuphold_scond_ccond {true | false}
```

Default: `false`

When set to `true`, enables proper generation of SDF (Standard Delay Format) with scond and ccond

qualifiers on `SETUPHOLD` and `RECREM` timing checks.

Since some SDF readers do not accept scond and ccond qualifiers this attribute is set to `false` by default.

timing_self_loop_paths_no_skew_max_depth

`timing_self_loop_paths_no_skew_max_depth` *value*

Default: 10

Identifies self-loop paths at the specified depth.

For example:

```
set_db timing_self_loop_paths_no_skew_max_depth 20
```

Allows the timer traverse 20 timing arcs (10 stages of logic) before abandoning the search for a self loop.

timing_self_loop_paths_no_skew_max_slack

`timing_self_loop_paths_no_skew_max_slack` *value*

Default: `0`

Identifies self-loop paths at the specified threshold.

For example:

```
set_db timing_self_loop_paths_no_skew_max_slack 1.0
```

In this case, all timing endpoints with a worst slack less than `1.0` are considered for self loop skew removal.

timing_set_clock_source_to_output_as_data

`timing_set_clock_source_to_output_as_data {true | false}`

Default: `false`

When set to `true`, causes a clock source path leading to an output or bidi port to be treated as a data path if there is `set_output_delay` or `set_data_check` assertion on the port.

timing_socv_preserve_variation_with_annotations

`timing_socv_preserve_variation_with_annotations {true | false}`

Default: `false`

Preserves the sigma value computed by delay calculation along with the annotation from SDC.

In the clock mesh flow, it is possible to preserve sigma values even after applying annotation on slew/delay through SDC.

When this attribute is set to `true`, the software accepts the sigma value computed by delay calculation (for nets that have delay/slew annotations from SDC, e.g., clock-mesh), along with the annotation from SDC.

timing_socv_rc_variation_mode

`timing_socv_rc_variation_mode  {true | false}`

Default: `false`

Enables interconnect variation mode in SOCV mode.

timing_socv_statistical_min_max_mode

`timing_socv_statistical_min_max_mode {statistical | mean_and_three_sigma_bounded}`

Default: `mean_and_three_sigma_bounded`

Determines how worst cases will be computed in SOCV mode.

You can select one of the following options:

- `statistical`: The mean and standard deviation for the result of worst-cases is computed to get the greatest or smallest of normal distributions.
- `mean_and_three_sigma_bounded`: The worst mean value is chosen as the mean of the resulting distribution. The standard deviation is computed as a bound from the worst (mean+/-3*sigma) value. This ensures that the 3-sigma value of the result always bounds the worst 3-sigma value.

  Note: The `mean_and_three_sigma_bounded` option is required for SOCV-based optimization in Innovus. It is also highly recommended for analysis in Tempus due to improved bounding properties.

timing_socv_view_based_nsigma_multiplier_mode

`timing_socv_view_based_nsigma_multiplier_mode {true | false}`

Default: `true`

Enables setting of view-based separate setup and hold sigma multipliers.

By default or with an explicit true setting, the timing system is configured to support different N-sigma multiplier values for Setup vs. Hold – and for different analysis views. The specific N-sigma multipliers can be set using the `set_socv_reporting_nsigma_multiplier` command.

timing_spatial_derate_chip_size

`timing_spatial_derate_chip_size` *`value_in_microns`*

Default: `1e+30`

Allows you to define the chip size.

The chip size is the diagonal length of a rectangle that is formed by the chip's length and width. You can obtain the diagonal distance by calculating the square-root of the $Length^2$ x $Width^2$. The diagonal distance should always be specified in units of microns.

The chip size can be picked in the following ways (in order of highest to lowest precedence):

- When set to a value other than `1e+30`, the specified value will have the highest precedence.
- If the physical floorplan information is available, the chip size will be derived automatically from the floorplan.
- If the SPEF parasitic files containing the coordinate information are loaded, then the chip size will be automatically derived.

timing_spatial_derate_distance_mode

`timing_spatial_derate_distance_mode {bounding_box | chip_size}`

Default: `bounding_box`

Allows you to select the graph-based analysis (GBA) spatial mode. The allowed values are - `bounding_box` or `chip_size`.

This attribute can be used when the spatial derate mode is enabled using the `timing_enable_spatial_derate_mode true` setting.

timing_enable_spatial_derate_mode

`timing_enable_spatial_derate_mode {true | false}`

Default: `false`

Enables the spatial derate mode.

When set to `true`, the software allows access to the GBA spatial modes (bounding_box or chip_size) specified with the `timing_spatial_derate_distance_mode` attribute.

When set to `false`, the software allows you to specify only the bounding_box option.

timing_suppress_escape_characters

`timing_suppress_escape_characters {true | false}`

Default: `true`

Suppresses the reporting of escape characters in timing object names.

When the attribute is set to `true`, all the reporting and collection commands report the timing objects names (e.g., pins, ports, or instances) without the escape characters.

timing_suppress_ilm_constraint_mismatches

`timing_suppress_ilm_constraint_mismatches {true | false}`

Default: `false`

When set to `true`, the software suppresses all error and warning messages related to objects not found when loading SDC constraint files for the ILM flow.

In the current ILM flow, SDC constraints originally specified against the complete flat netlist for the design do not get filtered when being applied against the ILM view of the design. Constraints that reference parts of the design that were pruned cause warnings and errors during constraint loading.

timing_time_unit

`timing_time_unit` *`string`*

Default: `" "`

Specifies the timing units to be used for timing constraints and timing reports. The default units for timing are inferred from the timing libraries. You can override this inference by explicitly setting the value to `none`, `1ns`, `1ps`, `10ps`, or, `100ps`. Setting this variable explicitly is the same as executing

the `set_library_unit -time` command to set the time units.

timing_use_clock_pin_attribute_for_clock_net_marking

`timing_use_clock_pin_attribute_for_clock_net_marking {true | false}`

Default: `false`

Allows the propagation of clocks to pins with clock attributes regardless of the presence of check arcs or trigger arcs at the pins.

timing_use_incremental_si_transition

`timing_use_incremental_si_transition {true | false}`

Default: `false`

Enables usage of incremental slew during DRV violation reporting.

timing_use_latch_early_launch_edge

`timing_use_latch_early_launch_edge {true | false}`

Default: `true`

Enables the latch use early launch edge feature.

During timing analysis when time borrowing and clock path pessimism removal (CPPR) is enabled, the given time is calculated using both borrow time and CPPR time. This may result in unnecessary timing violations. Since CPPR is an accurate means of pessimism removal, you are required to set the `timing_use_latch_early_launch_edge` attribute to `false` so that CPPR is applied to all the paths in the design.

When set to `true`, the launch clock path of timing paths starting at a latch is timed in early mode for setup analysis, and for late mode for hold analysis. No CPPR credit is given to the timing path.

When set to `false`, the behavior is like for any other path. The launch clock path of timing paths starting at a latch is timed in late mode for setup analysis, and early mode for hold analysis. The CPPR credit is given to the path according to the attribute CPPR settings.

timing_use_latch_time_borrow

`timing_use_latch_time_borrow {true | false}`

Default: `true`

Considers time borrowing during timing analysis. Time borrowing is the amount of time borrowed by a previous logic.

timing_use_verilog_for_model_netlist

```
timing_use_verilog_for_model_netlist
{true | false}
```

Default: `false`

Allows you to choose whether to write the scope and boundary model netlists in Verilog format instead of binary.

When set to `true`, the software will use Verilog format for saving model nets.

timing_waveform_aware_pulse_width_checks_high_voltage_level

```
timing_waveform_aware_pulse_width_checks_high_voltage_level double
```

Default: 0.95

Range: `{}`

Specifies the high voltage level to be used for waveform-aware pulse width checks.

timing_waveform_aware_pulse_width_checks_low_voltage_level

```
timing_waveform_aware_pulse_width_checks_low_voltage_level double
```

Default: `0.05`

Range: `{}`

Specifies the low voltage level to be used for waveform-aware pulse width checks.

timing_write_sdf_no_escape_backslash

```
timing_write_sdf_no_escape_backslash {true | false}
```

Default: `false`

Supports various patterns of the escape character.

For example, consider the following input Verilog expression:

\a/b\/c\/d

When the attribute is set to `true`, the `write_sdf` command removes the first escape character and outputs `a/b\/c\/d` in the SDF file.

When set to `false`, the command removes all the escape characters and outputs `a/b/c/d`.

update_analysis_view

```
update_analysis_view
[-help]
-name <existingAnalysisViewName>
{[-constraint_mode <newModeName>]
[-delay_corner <newDelayCornerObj>]
[-latency_file <string>]
[-power_modes <powerModesList>]}
```

Changes attribute values for the specified existing analysis view.

> You can update an analysis view before the multi-mode multi-corner definition file is loaded into the design, or after. However, after the software processes the file, any change to an existing object when the software is in multi-mode multi-corner mode results in the timing, delay calculation, and RC data being reset.

Parameters

`-constraint_mode` *newModeName*	Specifies the name of the new constraint mode to associate with this analysis view.
`-delay_corner` *newDelayCornerObj*	Specifies the name of the new delay calculation corner to associate with this analysis view.
`-latency_file` *string*	Specifies the latency file name.

`-name` *existingAnalysisViewName*	Specifies the name of the existing analysis view to update.
`-power_modes` *powerModesList*	Specifies list of power modes to use with this view. The modes must exist in the power intent file.

Examples

- The following command creates an analysis view called missionSlow using the constraint mode missionSetup and the delay calculation corner dcWCCOM:

```
create_analysis_view
-name missionSlow
-constraint_mode missionSetup
-delay_corner dcWCCOM
```

- The following command changes the delay calculation corner associated with the analysis view `missionSlow` to `dcMax`:

```
update_analysis_view -name missionSlow -delay_corner dcMax
```

Related Commands

- `create_analysis_view`
- `set_analysis_view`

update_constraint_mode

```
update_constraint_mode
-name existingConstraintModeName
{[[-sdc_files {newFile1.sdc newFile2.sdc...}]
[-ilm_sdc_files {newFile1.sdc newFile2.sdc ...}]
[-reset_latency_files]
[-tcl_variables {{var_name1 value1} {var_name2 value2} {var_name3 value3} ...} ]]}
```

Updates the SDC constraint file information for the specified existing constraint mode object.

Note: The `update_constraint_mode` command overwrites the previous constraint mode settings. However, by default, this command updates a constraint mode without affecting other constraint modes, including interactive constraints, such as, `set_dont_use` and `set_dont_touch`.

ⓘ You can update a constraint mode object before (or after) the multi-mode multi-corner definition file is loaded into the design. However, after the software processes the file, any change to an existing object when the software is in multi-mode multi-corner mode results in the timing, delay calculation, and RC data being reset.

Parameters

`-ilm_sdc_files {newFile1.sdc newFile2.sdc ...}`	Specifies the new SDC constraint files for the ILM flow to associate with this constraint mode.
`-name existingConstraintModeName`	Specifies the name of the existing constraint mode object to update.
`-sdc_files {newFile1.sdc newFile2.sdc ...}`	Specifies the new SDC constraint files to associate with this constraint mode.
`-reset_latency_files`	Cleanup latency files of all related views.
`-tcl_variables {{var_name1 value1} {var_name2 value2} {var_name3 value3} ...}`	
	Specifies a list of tcl variables and values to be used in the constraint mode.

Examples

- The following command groups the SDC files `io.sdc`, `mission1-clks.sdc`, and `mission1-except.sdc` to create a mode object named `missionSetup`:

```
create_constraint_mode -name missionSetup

-sdc_files [list io.sdc mission1-clks.sdc mission1-except.sdc]
```

- The following command changes the SDC files associated with the constraint mode `missionSetup` to `io.sdc`, `mission3-clks.sdc`, and `mission3-except.sdc`:

```
update_constraint_mode -name missionSetup

-sdc_files {io.sdc mission3-clks.sdc mission3-except.sdc}
```

Related Commands

- `create_constraint_mode`

update_delay_corner

```
update_delay_corner
[-help]
-name delayCornerName
[-early_estimated_worst_irDrop_factor percent]
[-early_irdrop_data files_or_directories]
[-early_irdrop_files list_of_files]
[-early_rc_corner rcCornerObj]
[-early_temperature_files list_of_files]
[-early_timing_condition conditional_pairs]
[-late_estimated_worst_irDrop_factor percent]
[-late_irdrop_data files_or_directories]
[-late_irdrop_files list_of_files]
[-late_rc_corner rcCornerObj]
[-late_temperature_files list_of_files]
[-late_timing_condition conditional_pairs]
[-irdrop_data files_or_directories]
[-irdrop_files list_of_files]
[-pg_net_voltages voltage_pairs]
[-rc_corner rcCornerObj]
[-si_enabled {true | false}]
[-temperature_files list_of_files]
[-timing_condition conditional_pairs]
```

Modifies the parameters of an existing delay calculation corner object. You can use the `update_delay_corner` command to add or change attribute values for an existing delay calculation corner object, or for an existing power domain definition.

You can update a delay calculation corner object before the multi-mode multi-corner definition file is loaded into the design, or after. However, after the software processes the file, any change to an existing object when the software is in multi-mode multi-corner mode results in the timing, delay calculation, and RC data being reset.

You also can use the `update_delay_corner` command to add a power domain definition to the specified delay calculation corner. Instances contained within an MSMV power domain often require a different library or operating condition that used for the default power domain. Power domain definitions within a delay calculation corner object can be used to assign specific libraries and PVT settings per power domain.
Note: Use separate delay calculation corners to define major PVT corner differences. Use the `-early_*` and `-late_*` parameters within a single delay calculation corner to control on-chip-variation.

The `-early_*` and `-late_*` parameters can be specified separately, to update a delay corner object

with early or late information. If you only specify one of the `-early_*` or `-late_*` parameters, the software uses the original information pointer in place of the missing early/late parameter.

For example, if you initially created a delay corner object using:

```
create_delay_corner
-library_set libsNominal
...
```

Use this command after creating at least one delay calculation corner (`create_delay_corner`).

Parameters

`-help`	Prints out the command usage.
`-early_estimated_worst_irDrop_factor` *per cent*	Specifies the early voltage derating factor. When this parameter is specified, the software performs delay calculations using the voltage = nomVolt * (1 – value). *Default*: 0.0
`-early_irdrop_data` *files_or_directories*	Specifies the IR drop files or directories to apply when calculating the early arrival times at a single delay corner.
`-early_irdrop_files` *list_of_files*	Specifies the list of IR drop files to apply when calculating early arrival times at a single power domain. This parameter is used primarily when configuring the multi-mode multi-corner environment for on-chip-variation (OCV) analysis. **Note**: This parameter is obsolete and will be removed in a future release. To ensure compatibility with future releases, you should update your scripts.
`-early_rc_corner` *rcCornerObj*	Specifies the RC corner object to associate with the early corner object.
`-early_temperature_files` *list_of_files*	Specifies the name of the temperature file for temperature aware delay calculation for early corner.

`-early_timing_condition` *conditional_pairs*	Specifies pair list of domains/supply sets and timing conditions to associate with this delay corner object.
`-irdrop_data` *files_or_directories*	Specifies the IR drop files or directories to apply to both the early and late delay calculation for the specified delay corner object.
`-irdrop_files` *list_of_files*	Specifies the list of IR drop files to apply to both early and late delay calculation for this power domain object. This parameter is used primarily to configure single-corner or Best-Case Worst-Case (BC-WC) analysis modes. **Note**: This parameter is obsolete and will be removed in a future release. To ensure compatibility with future releases, you should update your scripts.
`-late_estimated_worst_irDrop_factor` *percent*	Specifies the late voltage derating factor. When this parameter is specified, the software performs delay calculations using the voltage = nomVolt * (1 – value). *Default*: 0.0
`-late_irdrop_data` *files_or_directories*	Specifies the IR drop files or directories to apply when calculating the late arrival times at a single delay corner.
`-late_irdrop_files` *list_of_files*	Specifies the list of IR drop files to apply when calculating late arrival times at a single power domain. This parameter is used primarily when configuring the multi-mode multi-corner environment for on-chip-variation (OCV) analysis. **Note**: This parameter is obsolete and will be removed in a future release. To ensure compatibility with future releases, you should update your scripts.

`-late_temperature_files` *list_of_files*	Specifies the name of the temperature file for temperature aware delay calculation for late corner.
`-late_rc_corner` *rcCornerObj*	Specifies the RC corner object to associate with the late corner object.
`-late_timing_condition` *conditional_pairs*	Specifies pair list of domains/supply sets and timing conditions to associate with this delay corner object. **Note**: The `-early_timing_condition` and `-late_timing_condition` parameters should be specified together.
`-name` *delayCornerName*	Specifies the delay calculation corner to which to add the power domain definition, or to which to update the delay corner or power domain attribute values.
`-pg_net_voltages` *voltage_pairs*	Specifies the power/ground net voltages to be used for both early/late timing analysis.
`-rc_corner` *rcCornerObj*	Specifies the new RC corner object to associate with this delay corner.
`-si_enabled {true | false}`	Controls signal integrity (SI) analysis. When set to `true`, the analysis of all MMMC views will include SI analysis by default. You can use this parameter to further determine which analysis views receive SI analysis and which do not. When this parameter is set to `false` on a delay corner, the software will disable SI analysis for all the views that reference that corner. By default (or when this parameter is set to `true`), SI analysis is computed for all the views related to that corner - whenever SI is globally enabled. (The global control of SI analysis is determined by the `delaycal_enable_si true` attribute setting.)

`-temperature_files` *list_of_files*	Specifies the name of the temperature file for temperature aware delay calculation for single corner.
`-timing_condition` *conditional_pairs*	Specifies pair list of domains/supply sets and timing conditions to associate with this delay corner object.

Examples

- The following example specifies a list of conditional pairs that are domains/supply sets and the timing conditions associated with the specified delay corner object:

```
update_delay_corner -name ccworst_setup\
-timing_condition {tc_1 PD1@tc_2 PD2@tc_2 PD3@tc_2 PD4@tc_2}
```

Related Commands

- `create_delay_corner`

update_io_latency

```
update_io_latency
[-adjust_source_latency]
-verbose
```

Provides an automated means of balancing IO and Core clock latencies after the clock network has been added to the design so that pre and post clock tree skew between the clocks is maintained as closely as possible. This results in minimum change of inbound and outbound IO slacks.

This command performs the following actions:

- Measures and calculates ideal average latency for each Core clock.
- Calculates preCTS skew between IO and Core clocks.
- Asserts `set_propagated_clock` on all clock root pins or ports to put clock trees in propagated mode.
- Measures and calculates propagated mode average latency for each Core clock.
- Adjusts `set_clock_latency` on IO clocks network latency to minimize pre to post CTS IO-to-Core clock skew.

In case a single IO clock interacts with multiple Core clocks, the IO clock will be adjusted by the overall average latency of all the related Core clocks combined.

To be effective, this command should only be used after clock trees have been implemented and before the set_propagated_clock command is issued. This allows the command to see both the ideal and propagated state of the clock trees.

Parameters

`-adjust_source_latency`	Adjusts the source latency of core clock.
`-verbose`	Provides additional information to the console as adjustments are made to each IO clock.

update_library_set

```
update_library_set
[-help]
[-aocv string]
[-library_side_file string]
-name libSetName
[-si string]
[-socv string]
[-timing string]
```

Updates an existing specified library set.

Note: You can edit a library set before (or after) the multi-mode multi-corner view definition file is loaded into the design. However, after the software processes the file, any change to an existing object when the software is in multi-mode multi-corner mode resets the timing, delay calculation, and RC data.

Parameters

`-help`	Prints out the command usage.
`-aocv` *`string`*	Specifies a list of new AOCV libraries to include in the specified library set.
`-library_side_file` *`string`*	Specifies the library side file.
`-name` *`libSetName`*	Specifies the name of an existing library set to be updated. The library set must have been defined using the `create_library_set` command.
`-si` *`string`*	Specifies a list of new libraries and user-defined noise (UDN) models to include in the library set.
`-socv` *`string`*	Specifies a list of new SOCV libraries to be included in the library set.
`-timing` *`string`*	Specifies a list of new timing libraries to include in the specified library set. Note: The ECSM and CCS libraries can be very large and loading them may take significant time. You can convert the ASCII NLDM (Liberty .lib), ECSM, and CCS libraries to a binary version using the write_ldb command. For conversion of libraries to a binary version, refer to `write_ldb`. Instead of the original ASCII libraries, you can specify the following binary libraries in the configuration file: `create_library_set -name libName -timing "AAA.lib BBB.ecsm CCC.ccs"`

Examples

- The following command creates a library set that associates timing libraries with a nominal voltage of 1 volt with the library name lsCOM-1V:
  ```
  create_library_set -name IsCOM-1V -timing [list stdcell_F_1V.lib ram_F.lib pad.lib]
  ```
  The following command changes the timing libraries associated with the library set lsCOM-1V:
  ```
  update_library_set -name IsCOM-1V -timing [list stdcell_CVF_1V.lib ram_CVF.lib
  pad.lib ]
  ```

Related Commands

- create_library_set

update_opcond

```
update_opcond
[-help]
-name opcondName
[-process processValue]
[-temperature temperatureValue]
[-voltage voltageValue]
```

Updates a set of virtual operating conditions in the specified library that were created using the create_opcond command.

Parameters

-help	Prints out the command usage.
-name *opcondName*	Specifies the name of the operating condition to be updated.
-process *processValue*	Specifies the process value for the operating condition.
-temperature *temperatureValue*	Specifies the temperature value for the operating condition.
-voltage *voltageValue*	Specifies the voltage value for the operating condition.

Related Commands

- create_opcond

update_rc_corner

```
update_rc_corner
[-temperature temperatureValue]
[-cap_table newcapTableFile]
-name existingRcCornerName
[-post_route_res {resFactor1 resFactor2 resFactor3}]
[-post_route_cap {capFactor1 capFactor2 capFactor3}]
[-post_route_cross_cap {xcapFactor1 xcapFactor2 xcapFactor3}]
[-post_route_clock_res {resFactor1 resFactor2 resFactor3}]
[-post_route_clock_cap {capFactor1 capFactor2 capFactor3}]
[-pre_route_clock_res resFactor]
[-pre_route_clock_cap capFactor]
[-pre_route_res resFactor]
[-pre_route_cap capFactor]
[-qrc_tech newFileName]
[-via_variation_file viaVariationFile]
```

Adds or changes attribute values for the specified existing RC corner object.

> You can edit an RC corner object before the multi-mode multi-corner definition file is loaded into the design, or after. However, after the software processes the file, any change to an existing object when the software is in multi-mode multi-corner mode results in the timing, delay calculation, and RC data being reset.

You can set different postroute scale factors for the post-route engine variants by specifying them as duplets and triplets. For example,

resFactor1 resFactor2 resFactor3

Where:

- Single value: If you specify one value, the scale factor applies to effort level `low`. Scale factor value of 1 is used for `medium` and `high` by default.
- Duplet: If you specify two values, the first value is used for effort level `low` and the second value is used for `medium`. Scale factor value of `1` is used for `high` by default.
- Triplet: If you specify three values, the first value is used for effort level `low`, the second value is used for `medium`, and the third value is used for `high`.

Parameters

`-cap_table` *`newCapTableFile`*	Specifies a new capacitance table to use for extraction when using this RC corner. *Data_type*: string, optional
`-name` *`existingRcCornerName`*	Specifies the name of the existing RC corner object to update. *Data_type*: string, required
`-pre_route_cap` *`capFactor`*	Specifies the capacitance scale factor for RC Extraction in pre-route mode. *Data_type*: float, optional
`-pre_route_clock_cap` *`capFactor`*	Specifies the scale factor for capacitance of clock nets in pre-route mode. Default: `0` When this parameter is not specified, the value specified with the `-post_route_cap` parameter is used for pre-route extraction. When specified, this parameter overrides the value specified with the `-post_route_cap` parameter for clock nets. *Data_type*: float, optional
`-pre_route_clock_res` *`resFactor`*	Specifies the scale factor for resistance of clock nets in pre-route mode. Default: `0` When this parameter is not specified, the value specified with the `-post_route_res` parameter is used for pre-route extraction. When specified, this parameter overrides the value specified with the `-post_route_res` parameter for clock nets. *Data_type*: float, optional

`-pre_route_res` *resFactor*	Specifies the resistance scale factor for RC Extraction in pre-route mode. *Data_type*: float, optional
`-post_route_cap {`*capFactor1capFactor2 capFactor3*`}`	Specifies the capacitance scale factor for RC Extraction in post-route mode. *Data_type*: string, optional
`-post_route_clock_cap {`*capFactor1capFactor2 capFactor3*`}`	Specifies the scale factor for capacitance of clock nets in the post-route extraction mode. Default: `0` When this parameter is not specified, the value specified with the `-post_route_cap` parameter is used for post-route extraction. When specified, this parameter overrides the value specified with the `-post_route_cap` parameter. *Data_type*: string, optional
`-post_route_clock_res {`*resFactor1 resFactor2 resFactor3*`}`	Specifies the scale factor for resistance of clock nets in the post-route extraction mode. Default: `0` When this parameter is not specified, the value specified with the `-post_route_res` parameter is used for post-route extraction. When specified, this parameter overrides the value specified with the `-post_route_res` parameter for clock nets. *Data_type*: string, optional
`-post_route_cross_cap {`*xcapFactor1 xcapFactor2 xcapFactor3*`}`	Specifies the cross-coupling capacitance scale factor for RC extraction in post-route mode. *Data_type*: string, optional
`-post_route_res {`*resFactor1 resFactor2 resFactor3*`}`	Specifies the resistance scale factor for RC extraction in post-route mode. *Data_type*: string, optional

`-qrc_tech` *newFileName*	Specifies a new Quantus technology (`.tch`) file which is used to perform sign-off RC extraction. *Data_type*: string, optional
`-temperature` *newTempValue*	Specifies a new temperature, in units of Celsius, to use to derate resistance values. *Data_type*: float, optional

`-via_variation_file` *viaVariationFile*	Specifies a file for modeling via variation. The VIA variation file lists up all the VIA layer names with the corresponding multiplier. Each RC corner can have its own VIA variation file.
	For example:
	`update_rc_corner`
	`-rc_corner rcCornerName1 \`
	`-via_variation_file viaVariationFileName1`
	`update_rc_corner`
	`-rc_corner rcCornerName2 \`
	`-via_variation_file viaVariationFileName2`

Examples

- The following command creates an RC corner called `rc-cbest` that uses the capacitance table `myTech_bc.CapTbl`, and derates the resistance values based on the temperature of 50 Celsius:
  `create_rc_corner -name rc-cbest -cap_table myTech.4_bc.CapTbl -temperature 50`
- The following command changes the capacitance table used by RC corner `rc-cbest` to `myCap.2_bc.capTbl`, and the temperature by which resistance values are derated to `25` Celsius:
  `update_rc_corner -name rc_cbest -temperature 25`

Related Information

- `create_rc_corner`
- "Design Import and Export in Stylus" chapter in the *Innovus Stylus Common UI User Guide*

- Configuring the Setup for Multi-Mode Multi-Corner Analysis
- Creating RC Corner Objects

Related Commands

- create_rc_corner

update_timing_condition

```
update_timing_condition
[-help]
[-library_sets string]
-name timingConditionName
[-opcond string]
[-opcond_library string]
```

Updates an existing timing condition with new operating conditions or library sets.

This command performs an update on the specified timing condition. The named condition must be present, or an error is issued. The options completely replace the contents of the specified attribute.

Parameters

`-help`	Prints out the command usage. For a detailed description of the command and all of its parameters, use the `man` command: `man update_timing_condition`.
`-library_sets string`	Specifies a list of library sets to associate with the timing condition.
`-name timingConditionName`	Specifies a unique name for the timing condition. This name must exist or an error is displayed.
`-opcond string`	Specifies a single name for the operating condition to associate with the timing condition.
`-opcond_library string`	Specifies the name of library to search for the named operating condition.

Examples

- The following example shows the `update_timing_condition` command performs an update on the specified timing condition:

  ```
  create_timing_condition -name TC1 -library_sets {ls1 ls2} -opcond oc1
  update_timing_condition -name TC1 -library_sets {ls3}; # TC1 now only has libset ls3
  ```

- The following commands result in identical results. However, the first command creates the bin TC2 and the second command updates both the options:

```
create_timing_condition -name TC2 -library_sets {ls1 ls2} -opcond oc1
update_timing_condition -name TC2 -library_sets {ls1 ls2} -opcond oc1
```

Related Commands

- create_timing_condition

write_combinational_loop_break_constraints

```
write_combinational_loop_break_constraints
-in_file fileName
out_file fileName
```

Generates SDC file containing constraints for combinational loop breaking. The command selects the disable arc (s) with flag set to 1 from the report file, and generates the equivalent SDC commands in an output file.

You can load the output SDC file at any time during an Innovus System session. To load the file, include it in the `.config` file.

Parameters

`-in_file` *fileName*	Specifies the disabled time arcs report that you generate. To generate the report, use the following command: `report_disbale_timing -nosplit > disable_arcs.rpt`
`out_file` *fileName*	Specifies the name of the output SDC file.

Command Order

Use this command after generating the disable arcs report.

Example

The following command generates an `encCombLoopBreak.sdc` file when the input is a `pt_disable_arcs.rpt` file:

```
write_combinational_loop_break_constraints -in_file disable_arcs.rpt
            out_file encCombLoopBreak.sdc
```

The contents of `pt_disable_arcs.rpt` file are as follows:

Cell or Port	From To Sense Flag Reason
----------	
uchip_core/u_s5_power	oLpcSd_4_ orig_oSmAcpiSd_4_ * 1

The contents of output file `encCombLoopBreak.sdc` are as follows:

```
:
set_disable_timing -from {oLpcSd_4_} -to {orig_oSmAcpiSd_4_}
                [get_cells {uchip_core/u_s5_power}]:
```

write_global_slack_report

```
write_global_slack_report
[-help]
[-early]
[-late]
[-rise]
[-fall]
[-include_ports]
[-max_slack val]
[-min_slack val]
[-view view_name]
[> | >> file_name]
[pin_port_list]
```

Reports early and late slacks on every instance pin in the design.

This command computes and reports the required time and slack values of the specified pins.

Parameters

-help	Prints out the command usage.
> | >> *filename*	Specifies the output file name. When specified with >, writes the report to the specified file name. If the file already exists, the software overwrites it. When specified with >>, writes the report to the specified file name. If the file already exists, the software concatenates the report to the end of the file. **Note:** To write a compressed report, add the .gz extension to the file name.
-early/-late	Specifies the mode for which slack is to be reported.
-fall/-rise	Specifies the transition for which slack is to be reported.

`-include_ports`	Reports slack information for ports. By default, the slack information is reported for pins only.
`-max_slack` *val*	Reports pins that have early and late slack values less than the specified value. Type: Float
`-min_slack` *val*	Reports pins that have early and late slack values less than the specified value. *Type:* `Float` **Note**: This parameter is obsolete and will be removed in a future release. Cadence recommends that you use the `-max_slack` parameter.
pin_port_list	Specifies the list of pins or ports for which slack information is to be reported.
`-view` *view_name*	Reports slack information for the specified view.

Examples

- The following command reports the early and late slacks on every instance pin in the design:

```
> write_global_slack_report > out.rpt

# Unit: 1ns
# <late_rise> <late_fall> <early_rise> <early_fall> <view_name> <hpin_name>
# ------------------------------------------------------------------------
*              *          *            *            (null)      u2/Q
*              *          *            *            (null)      u8/Q
*              *          *            *            (null)      b11/A
*              *          *            *            (null)      b11/Y
0.823          0.347      0.115        0.501        (null)      u10999/A
0.823          0.347      0.115        0.501        (null)      u10999/Y
# ------------------------------------------------------------------------
```

- The following command reports slack for a pin collection:

```
> set x [get_pins -hier */A]
```

```
0x14 <---- 'x' stores Pointer for pin collection
```

- The following command will generate a global slack report for pin collection as defined in "x":

```
> write_global_slack_report $x


# Unit: 1ns
# <late_rise> <late_fall>  <early_rise>  <early_fall>  <view_name> <hpin_name>
# ------------------------------------------------------------------------
19.622       19.742        0.321         0.106         (null)      seg1/u1/A
19.622       19.742        0.321         0.106         (null)      seg1/u2/A
18.018       18.362        1.876         1.486         (null)      seg1/u10/A
18.018       18.362        1.876         1.486         (null)      seg1/u11/A
18.362       18.018        1.486         1.876         (null)      seg1/u12/A
18.018       18.362        1.876         1.486         (null)      seg1/u13/A
25.845       25.883       -0.934        -1.053         (null)      seg2/u1/A
25.845       25.883       -0.934        -1.053         (null)      seg2/u2/A
# ------------------------------------------------------------------------
```

- The following example shows slack calculations, when you run the `report_timing` command for the same hpin.

```
> write_global_slack_report DT_IN/RE_CO/r697_reg_2/D

# Unit: 1ns
# <late_rise> <view_name> <late_fall> <view_name> <early_rise>
<view_name> <early_fall> <view_name> <hpin_name>
# ------------------------------------------------------------------------
12.641 dtmf_view_setup 12.127 dtmf_view_setup * *
DT_IN/RE_CO/r697_reg_2/D

> report_timing -to DT_IN/RE_CO/r697_reg_2/D -late -fall

Path 1: MET Setup Check with Pin DT_IN/RE_CO/r697_reg_2/CKN
Endpoint: DT_IN/RE_CO/r697_reg_2/D (v) checked with trailing
edge of 'vclk2'
Beginpoint: DT_IN/RE_CO/r697_reg_2/Q (v) triggered by trailing
edge of 'vclk2'
Path Groups: {vclk2}
Analysis View: dtmf_view_setup
Other End Arrival Time 7.015
- Setup 0.600
+ Phase Shift 14.000
- Uncertainty 0.250
= Required Time 20.164
```

```
- Arrival Time 8.037
= Slack Time 12.127
Clock Fall Edge 7.000
+ Clock Network Latency (Prop) 0.015
= Beginpoint Arrival Time 7.015
```

Instance	Arc	Cell	Delay	Arrival Time	Required Time
DT_IN/RE_CO/r697_reg_2	CKN v			7.015	19.142
DT_IN/RE_CO/r697_reg_2	CKN v -> Q v	SDFFNX1	0.710	7.725	19.852
DT_IN/RE_CO/i_6594	B v -> Y v	MX2X1	0.312	8.037	20.164
DT_IN/RE_CO/r697_reg_2	D v	SDFFNX1	0.000	8.037	20.164

```
> report_timing -to DT_IN/RE_CO/r697_reg_2/D -late -rise


Path 1: MET Setup Check with Pin DT_IN/RE_CO/r697_reg_2/CKN
Endpoint: DT_IN/RE_CO/r697_reg_2/D (^) checked with trailing
edge of 'vclk2'
Beginpoint: DT_IN/RE_CO/r697_reg_2/Q (^) triggered by trailing
edge of 'vclk2'
Path Groups: {vclk2}
Analysis View: dtmf_view_setup
Other End Arrival Time 7.015
- Setup 0.204
+ Phase Shift 14.000
- Uncertainty 0.250
= Required Time 20.560
- Arrival Time 7.919
= Slack Time 12.641
Clock Fall Edge 7.000
+ Clock Network Latency (Prop) 0.015
= Beginpoint Arrival Time 7.015
```

Instance	Arc	Cell	Delay	Arrival Time	Required Time
DT_IN/RE_CO/r697_reg_2	CKN v			7.015	19.656
DT_IN/RE_CO/r697_reg_2	CKN v -> Q ^	SDFFNX1	0.628	7.643	20.284
DT_IN/RE_CO/i_6594	B ^ -> Y ^	MX2X1	0.276	7.919	20.560
DT_IN/RE_CO/r697_reg_2	D ^	SDFFNX1	0.000	7.919	20.560

write_ldb

```
write_ldb
[-help]
ldb_file
-library string
```

Reads and generates compiled libraries.

You can use the `write_ldb` command to run timing and signal integrity analysis with binary timing libraries.

The ECSM and CCS libraries can be very large and loading them may take significant time. You can convert the ASCII NLDM (Liberty .lib), ECSM, and CCS libraries to a binary version using the `write_ldb` command. When loading the design, the reading of the timing libraries will be faster because it reads the binary versions.

The following example illustrates the conversion of libraries to a binary version:

```
write_ldb -library AAA.lib ldb_file AAA.db
write_ldb -library BBB.ecsm ldb_file BBB.db
write_ldb -library CCC.ccs ldb_file CCC.db
```

Instead of the original ASCII libraries, you can specify these binary libraries in the configuration file:

- **Original MMMC setup file**: `create_library_set -name libName -timing "AAA.lib BBB.ecsm CCC.ccs"`
- **New MMMC setup file**: `create_library_set -name libName -timing "AAA.db BBB.db CCC.db"`

Notes

- The software does not support concatenated Liberty library (multiple libraries in the same `.lib` file) files during library database (LDB) conversion. To use these files, you are required to separate out individual libraries before converting to individual `.ldb` files. This feature may be available in future releases.
- The software does not support library database conversion for Timing Library Format (TLF) files.

Parameters

`-library` *string*	Specifies the name of the input library file.
ldb_file	Specifies the name of the generated compiled library file.

The input library file can also be an old compiled library. In this case the output compiled library file will be in the latest compiled version.

Related Commands

- `check_ldb_version`

write_path_descriptions

```
write_path_descriptions
[-help]
var_name
-out_file string
```

Writes out the path description of all the specified timing paths to an ASCII file. The output file contains a set of `report_timing` commands for each timing path in the given path collection. The file also includes a complete description of each timing path captured through timing points along with the clock and edge information.

Parameters

`-help`	Prints out the command usage.
var_name	Specifies the object collection.
`-out_file` *string*	Redirects output to the specified file.

Examples

- The following example command writes out descriptions of late GBA paths to `pathfile1`:

  ```
  write_path_descriptions var1 -out_file file1 > file2
  ```

  The output file `pathfile1` will contain the following information:

  ```
  report_timing -net -path_type full_clock -late -from_rise A -through_rise B -
  through_rise C -to_rise D -view slow_v -clock_from clk1 -edge_from lead -clock_to
  clk2 -edge_to lead
  ```

Related Commands

- `read_path_descriptions`

write_sdc

```
write_sdc
[-help]
[file_name]
[-exclude_annotated <string>]
[-path_adjust_only]
[-path_group_only]
[-sdc]
[-view <string>]
```

Generates the timing constraints file in a third-party compiler format.

Parameters

`-help`	Prints out the command usage.
`file_name`	Specifies the name of the constraints file. By default, the constraint file is generated in the current directory. To save to a different directory, you need to specify the complete path.
`-exclude_annotated` *`string`*	Generates the constraints file excluding annotated constraints. You can specify one of the following options: • `delays` • `checks` • `transitions`
`-path_adjust_only`	Exports `set_path_adjust_group` and `set_path_adjust` constraints.
`-path_group_only`	Exports `group_path` constraints.
`-strict_sdc`	Produces a SDC file where functions to manipulate collections are expanded. For example, the `remove_from_collection all_inputs`, `all_outputs`, `all_clocks`, and filter commands are replaced by the resulting list of objects. **Note:** When you specify the `-strict_sdc` parameter, a warning is issued stating that using the parameter produces a constraint file containing SDC standard constraints only. This may cause some design constraints to be dropped.
`-view` *`viewName`*	Specifies a view name to obtain the desired SDC.

Command Order

Use this command after reading in a timing constraints file.

Examples

- The following command generates constraints file `newc` and saves the generated file in the directory `new`:

  ```
  write_sdc new/newc
  ```

- Specifying the following constraints will have new entries in the output constraints file:

  ```
  set_annotated_delay -from u1/A -to u1/Y -cell 2
  set_annotated_check 2 -setup -from f0/CK -to f0/D
  ```

  The `write_sdc` output will be as follows:

  ```
  set_annotated_delay -cell 2 -from [get_pins {u1/A}] -to [get_pins {u1/Y}]
  set_annotated_check -setup -from [get_pins {f0/CK}] -to [get_pins {f0/D}] 2
  ```

- For the following constraints in the design :

  ```
  set_annotated_delay -from u1/A -to u1/Y -cell 2
  set_annotated_transition 2 u1/A
  set_annotated_check 2 -setup -from f0/CK -to f0/D
  ```

  The `write_sdc -exclude_annotated delays` output will be as follows:

  ```
  set_annotated_transition 2 [get_pins {u1/A}]
  set_annotated_check -setup -from [get_pins {f0/CK}] -to [get_pins {f0/D}] 2
  ```

  The `write_sdc -exclude_annotated {delays transitions}` command outputs the following:

  ```
  set_annotated_check -setup -from [get_pins {f0/CK}] to [get_pins {f0/D}] 2
  ```

write_sdf

```
write_sdf
[-help]
out_file
[-adjust_setup_hold_for_zero_hold_slack]
[-base_delay]
[-cell_timing {all | none | nochecks}]
[-collapse_internal_pins]
[-delimiter char]
[-delta_delay]
[-edges {edged | library | noedge | check_edge}]
[-exclude_cells cell_list]
[-exclude_cells_keep_io_nets]
[-exclude_cells_keep_top_level_nets]
[-exclude_disabled_arcs]
[-exclude_disabled_modes]
[-exclude_whatif_arcs]
[-ideal_clock_network]
[-include_condelse]
[-interconnect {all | none | noport | noinport | nooutport}]
[-map_file {file1 file2 ...}]
[-map_internal_arcs]
[-map_negative_checks]
[-map_negative_delays]
[-map_out_to_out_arcs]
[-map_recrem {merge_always | split | merge_when_paired}]
[-map_removal]
[-map_setuphold {merge_always | split | merge_when_paired}]
[-max_view viewName]
[-min_period_edges {posedge | negedge | both | none}]
[-min_view viewName]
[-no_condition]
[-no_derate]
[-no_escape]
[-no_variation]
[-precision non_neg_integer]
[-process string]
[-recompute_delay_calc]
[-recompute_parallel_arcs]
[-scale float]
[-sort_check_values {order}]
[-sort_delay_values]
```

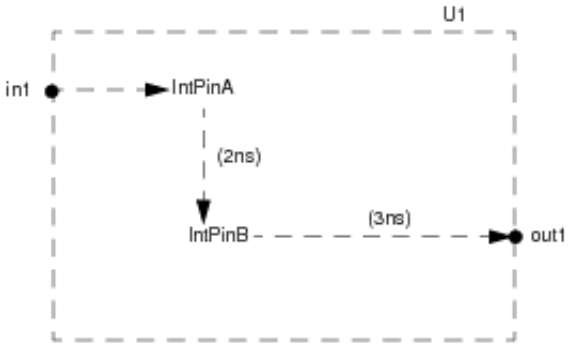
```
[-target_application {sta | verilog}]
[-gate_level_sim_model]
[-temperature string]
[-typical_view viewName]
[-version {2.1 | 3.0}]
[-view viewName]
[-voltage string]
```

Writes delays to a Standard Delay Format (SDF) file. The `write_sdf` command can generate both plain-text and GNU zipped (`.gz`) SDF files.

By default, the `write_sdf` command uses the delays already cached by the system and writes only values of the triplets.

Parameters

`-help`	Prints out the command usage.
out_file	Specifies the name of the SDF output file. If omitted, this argument defaults to `stdout`.
`-adjust_setup_hold_for_zero_hold_slack`	Adjusts hold timing check to meet non-negative slack.
`-base_delay`	Dumps base delay into SDF file.
`-cell_timing {all | none | nochecks}`	Specifies which cell delays and timing checks to write out. *Default*: `all` • `all` - Writes all cell delays and timing checks to the SDF file. This is the default. • `nochecks` - Disables the writing of timing checks only. • `none` - Prevents cell delays and timing checks from being written into the SDF file.

`-collapse_internal_pins`	Collapses arcs across internal pins to a single primary input to primary output arc.
	
	Use the -collapse_internal_pins parameter when the SDF file being generated is intended for use by an application that does not support the same internal pin structure in its leaf cell libraries (for example, Cadence VXL and NCsim Verilog simulation products). Do *not* use this parameter to generate an SDF file intended for use within the flow, or by third party tools that utilize the same Liberty timing models.
	
	Default: Arcs between internal pins in a timing model are written to the SDF file as explicit annotations to and from the internal pins.
	
	For example, assume the following arc between internal pins:
	
	 
	
	The following SDF would be generated by default:
	
	`(U1/in1 U1/IntPinA (1::1) (1::1))`
	
	`(U1/IntPinA U1/IntPinB (2::2) (2::2))`
	
	`(U1/IntPinB U1/out1 (3::3) (3::3))`
	
	If you specify -collapse_internal_pins, the resulting SDF would show a single arc with the merged delays:
	
	`(U1/in1 U1/out1 (6::6) (6::6))`

`-delimiter` *char*	Specifies the hierarchy divider character to use in the output file. This option sets the hierarchy divider of only the `write_sdf` command. If omitted, the default hierarchy divider character (`/`) is used as the `write_sdf` default.
`-delta_delay`	Writes out delta delay values to the SDF file.
`-edges {edged | library | noedge | check_edge}`	Specifies the edges values. Refer to Table for full qualifications of the possible values for the edges option. *Default*: `edged` • `check_edge` - Keeps edge specifiers on timing check arcs but does not add edge specifiers on delay arcs. • `edged` - Keeps edge specifiers on timing check arcs and delay arcs. • `library` - Assumes that the sdf_edges attributes are specified in the .lib file. The default sdf_edges value of a timing arc is noedge. • `noedge` - Defined in Table.
`-exclude_cells` *cell_list*	Specifies a list of leaf-cell or hierarchical instances whose timing data will be excluded from the the SDF (Standard Delay Format) file. If you specify a hierarchical instance, the timing data for all leaf-cells below the level of hierarchy will be omitted from the SDF file.
`-exclude_cells_keep_io_nets`	Restricts the filtering to annotations that are completely enclosed by the cell. Annotations to or from the cell will be preserved. By default, when the `-exclude_cells` option is used, any annotations within the excluded cell and any annotations to or from the excluded cell are filtered from the SDF.

`-exclude_cells_keep_top_level_nets`	Allows you to configure the handling of top-level interconnect annotations to an instance that is excluded from the SDF using the `write_sdf -exclude_cells` parameter. When this parameter is specified, the interconnect arcs which span to the ports of the instance will be included in the SDF. By default, no interconnect arcs to the instance are included. The delay annotated will be the worst-case delays of the top-level nets to/from the first level sink/source pins within the block that traverse a given input/output port of the instance, respectively.
`-exclude_disabled_arcs`	Excludes user set disabled arcs in the Standard Delay Format (SDF) file. If the user sets this option, the disabled arcs set by commands `set_mode`, `set_disable_timing`, and `set_case_analysis` are excluded.
`-exclude_disabled_modes`	Excludes timing arcs that have been disabled due to an active library mode set by the `set_mode` command, from appearing in the SDF file.
`-exclude_whatif_arcs`	Excludes what-if arcs in the SDF file.
`-ideal_clock_network`	Marks delays of delay arcs, that are on the ideal clock networks, as 0 in the SDF file.
`-include_condelse`	Writes `CONDELSE` constructs with the default value when a `COND` construct is written. **Note:** Use this argument with the `-version 3.0` option.

`-interconnect {all | none | noport | noinport | nooutport}`	Specifies which interconnect delays to write. *Default*: `all` • `all` - Writes all `INTERCONN` delays into the SDF file. This is the default if the `-interconnect` option is not specified. • `none` - No `INTERCONN` delays are written into the SDF file. • `noport` - No top level port `INTERCONN` delays are written into the SDF file. • `nooutport` - Disables only the output port `INTERCONN` delays.
`-map_file {file1 file2 ...}`	Specifies a list of map files. The map files can be used to determine what is written into the SDF files. For example, a map file can provide delay information on internal pins which are not defined in the timing library. Syntax for a map file (for a cell) is as follows: • A template section (the text to output) that reference variables of the form $<number> • A variable definition section that tells the expression to evaluate to obtain the value
`-map_internal_arcs`	Merges two arcs that are related to the same internal pin and forms one timing arc.
`-map_negative_checks`	Converts all negative timing check values to 0.0.
`-map_negative_delays`	Converts all negative IOPATH delays to zero.
`-map_out_to_out_arcs`	Writes out all output-to-output timing arcs as input-to-output arcs (bool, optional)
`-map_recrem {merge_always | split | merge_when_paired}`	Determines how RECOVERY and REMOVAL timing checks are written.

`-map_removal`	Converts the SDF v2.1 unsupported REMOVAL to HOLD checks when used with the -version 2.1 option.
`-map_setuphold {merge_always | split | merge_when_paired}`	Determines how SETUP and HOLD timing checks are written.
`-max_view` *viewName*	Uses the late delay from the specified analysis view to populate the SDF max slot, when in multi-mode multi-corner analysis mode.
`-min_period_edges {posedge | negedge | both | none}`	Specifies how to write out minimum `PERIOD` checks. Two check arcs are always created for the `min_period` check when you read in a `.lib` file. The following options let you keep or remove the positive edge, the negative edge, both edges, or no edges from the PERIOD statement in the `.sdf` file. • `posedge` - Writes out one positive edge `PERIOD` check. • `negedge` - Writes out one negative edge `PERIOD` check. • `both` - Writes out both the positive edge and the negative edge `PERIOD` checks. This is the default. • `none` - Writes out one `PERIOD` check without any edge specifier.
`-min_view` *viewName*	Uses the early delay from the specified analysis view to populate the SDF `min` slot, when in multi-mode multi-corner analysis mode.
`-no_condition`	Suppresses the `COND` statements in the SDF file.

`-no_derate`	Writes non-derated delays to the Standard Delay Format (SDF) file. By default, the `set_timing_derate` command writes scaled delays to the SDF file. When the `-no_derate` parameter is specified, the `write_sdf` command removes all the applied derates.
`-no_escape`	Writes out object names without escaping hierarchical dividers. Normally, when writing out an SDF, the hierarchical divider (`/`) is escaped because it is a special character. You can set `timing_write_sdf_no_escape_backslash` timing attribute to control the handling of escape characters when `write_sdf -no_escape` parameter is specified.
`-no_variation`	Exports the SDF delay and check values without including the variation component, when the selected analysis type is SOCV. By default, the SDF delays and check values will have the mean + N*sigma value.
`-precision` *non_neg_integer*	Controls the number of digits appearing after the decimal point in the output SDF file. The default precision is the value of the `timing_report_precision`.

`-process` *string*	Specifies one or a combination of the following process names that will appear in the header of the generated SDF file:
	• `best`
	• `typical`
	• `worst`
	Use the colon delimiter to specify multiple process names. For example:
	`-process best:typical:worst`
	In the above example, the header of the generated SDF file will contain:
	`(PROCESS "best:typical:worst")`

`-recompute_delay_calc`	Instructs the software to recompute any necessary data required for a complete SDF file before exporting.

`-recompute_parallel_arcs`	Instructs the software to temporarily recompute delays with separate delay calculation for parallel arcs. The software normally operates in a high performance mode which optimizes the handling of multiple timing arcs between input and output pin pairs. By default, the SDF file generated will contain the same delay information for all of these arcs. To have the SDF recalculated with explicit pin pair data, you can use the `-recompute_parallel_arcs` parameter to cause delays for each parallel arc to be explicitly calculated. It is recommended to use this parameter for generating SDF for functional simulation applications. **Note**: When you use the `write_sdf -recompute_parallel_arcs` option timing analysis is performed again. You need to obtain a new delay calculation to generate the SDF, and another delay calculation to put the system back to where it was before `write_sdf` was specified. Since delay calculation is run again, timing analysis will be reset. **Note**: The `-recompute_parallel_arcs` parameter is obsolete. This parameter still works in this release, but to ensure compatibility with future releases update your scripts to use the `-recompute_delay_calc` parameter instead.
`-scale` *float*	Specifies a multiplier to the delay values used in the SDF file. This option does not change the `timescale` setting of the SDF file. The delay values are scaled, but the units are the same. *Default*: `1`
`-sort_check_values {order}`	Sorts the check delays in OCV analysis.
`-sort_delay_values`	Sorts values in the SDF delay triplets into ascending order.

`-target_application {sta | verilog}`	Allows customization of the SDF output for use with STA or Verilog applications. You can select the following options: • sta: You can use the sta option to configure the SDF output for use with static-timing analysis (STA) applications. This includes sign-off timing analysis with products, such as Cadence Tempus, as well as for working with implementation flows that use STA for internal timing analysis. The sta option preserves maximum accuracy and generates SDF output that matches the structure of the Liberty timing libraries. • verilog: You can use the verilog option to generate SDF output that is compatible with Cadence Verilog-XL or Verilog-NC simulators. When the `-target_application` option is not specified then by default SDF output is configured for use with STA applications.
`-gate_level_sim_model`	Writes out the GLS SDF file.

`-temperature` *string*	Defines the temperature values that will appear in the header of the generated SDF file.
	Use the colon delimiter to specify multiple temperature values. For example:
	`-temperature -40.0:25.0:125`
	In the above example, the header of the generated SDF file will contain:
	`(TEMPERATURE -40.0:25.0:125)`

`-typical_view` *viewName*	Uses the late delay from the specified analysis view to populate the SDF `typ` slot, when in multi-mode multi-corner analysis mode.
`-version {2.1 | 3.0}`	Specifies whether to generate the SDF V2.1 or V3.0. The default SDF version is 3.0.

`-view` *viewName*	Uses the early and late delays for the specified analysis view to populate the `min` and `max` SDF triplet slots, when in multi-mode multi-corner analysis mode. The SDF `typ` slot remains empty (*earlyVal*`::`*lateVal*).

`-voltage` *string*	Defines the voltage values that will appear in the header of the generated SDF file.
	Use the colon delimiter to specify multiple voltage values. For example:
	`-voltage 1.980:1.800:1.620`
	In the above example, the header of the generated SDF file will contain:
	`(VOLTAGE 1.980:1.800:1.620)`

Table: **Definition of -edges Option Values**

	edged Value	**library Value**	**noedge Value**
IOPATH from a unate input pin to output pin	No edge specifier	No edge specifier	No edge specifier
IOPATH from a non-unate input pin to output pin	Edge specifier at the input pin	Edge specifier at the input pin when sdf_edges is either "both_edges" or "start_edge"	No edge specifier When merging edged paths, the maximum values are used for MAX and TYP fields; minimum values are used for MIN fields.

IOPATH of a clock/enable pin to data out pin	Edge specifier at the input pin	Edge specifier at the input pin when sdf_edges is either "both_edges" or "start_edge"	No edge specifier
Timing Checks	Edge specifiers match the check arcs specified in the timing library	Overridden by value of the sdf_edges attribute For both_edges, the timing checks will contain edge specifier on both starting and ending pins For start_edge, the starting pin (for HOLD type check) or ending pin (for SETUP type check) is edge-specified For end_edge, the ending pin (for HOLD style check) or the starting pin (for SETUP type check) is edge-specified The timing checks are duplicated or combined appropriately. The start_edge	Only the edge specifiers of the reference pin are written

		and end_edge are interpreted as first and second events, respectively, of the Verilog-XL's $hold() and $setup system tasks	

Examples

- The following command writes out an SDF 2.1 compliant output file named `my.sdf` with 4 digits after the decimal point: `write_sdf -version 2.1 -precision 4 my.sdf`
- The following example SDF files compare the output of `-edges edged` to the output of `-edges noedge`.

  Resultant SDF with `-edges` set to `edged`:

  ```
  (HOLD (posedge D) (posedge CK) (-0.14:-0.14:-0.14)) (HOLD (negedge D) (posedge CK) (-0.06:-0.06:-0.06)) (SETUP (posedge D) (posedge CK) (0.22:0.22:0.22)) (SETUP (negedge D) (posedge CK) (0.41:0.41:0.41))
  ```

  With `-edges` set to `noedge` only the edge specifiers of the clock pin are written.

  ```
  (HOLD D (posedge CK) (-0.14:-0.06:-0.06)) (SETUP D (posedge CK) (0.22:0.41:0.41))
  ```

Related Commands

- `read_sdf`

write_timing_windows

```
write_timing_windows
[-pin]
[-power_compatible]
[-view string]
[-ssta_sigma_multiplier float]
[-voltage_threshold string]
out_file
```

Generates a timing window file (TWF). The TWF file mainly contains the earliest and the latest possible arrival times that a signal may arrive on a net or a pin.

The format of a timing window file consists of eight pairs of data as given below:

```
(NET <netname>

minRTW:maxRTW minRt:maxRt minRDr:maxRDr minRSlk:maxRSlk minFTW:maxFTW minFt:maxFt
minFDr:maxFDr minFSlk:maxFSlk)
```

Where:

- `minRTW`: Minimum rise arrival time
- `maxRTW`: Maximum rise arrival time
- `minRt`: Minimum rise transition time
- `maxRt`: Maximum rise transition time
- `minRDr`: Minimum rise source resistance
- `maxRDr`: Maximum rise source resistance
- `minRSlk`: Minimum rise slack time
- `maxRSlk`: Maximum rise slack time
- `minFTW`: Minimum fall timing window
- `maxFTW`: Maximum fall timing window
- `minFt`: Minimum fall transition time
- `maxFt`: Minimum fall transition time
- `minFDr`: Minimum fall source resistance
- `maxFDr`: Maximum fall source resistance

- `minFSlk`: Minimum fall slack time
- `maxFSlk`: Maximum fall slack time

Parameters

`-help`	Prints out the command usage.
`out_file`	Writes the TWF file to the specified file name. The *`filename`* parameter must be the last argument in the list. **Note:** To write a compressed file, add the `.gz` extension to the file name.
`-pin`	Generates a TWF for each pin instead of each net. *Default*: Generates a TWF for the crosstalk analysis tool, which contains timing windows on each net.
`-power_compatible`	Generates a timing window file that is suitable for power analysis.
`-ssta_sigma_multiplier` *`float`*	Specifies a sigma multiplier between `0` and `3`. The value that you specify increases the arrival window by that amount and worst slack value is generated. The transition and impedance values are not impacted.
`-view` *`string`*	Generates a TWF for the specified multi-mode multi-corner view only. You must be multi-mode multi-corner analysis mode to use this parameter. If the `-view` parameter is not specified, then by default the first active view will be used to generate timing windows.
`-voltage_threshold` *`string`*	Specifies voltage threshold values for generating a Timing Window File (TWF). All the slews in the TWF file will be scaled with these values. The lower threshold value must be less than the higher threshold value. *Value range:* `0` to `100`

Examples

- The `write_timing_windows -voltage_threshold {20 80}` command outputs the following in the TWF file:

  ```
  (VOLTAGE_THRESHOLD 20 80)
  ```

- When active clocks are set using the `set_active_clocks` command, the `write_timing_windows` command will not display windows for inactive clocks in the output. The example report is shown below:

```
(TIMING_WINDOWS
(HEADER
(VERSION "0.1")
(DATE "Tue Apr 2 23:07:58 2013")
(DESIGN "top")
(DELIMITERS "/[]")
(VOLTAGE_THRESHOLD 20 80)
(PROCESS 1.000 1.000 1.000)
(VOLTAGE 1.012 1.012 1.012)
(TEMPERATURE 125.000 125.000 125.000)
(OPERATING_CONDITIONS DEFAULT DEFAULT DEFAULT)
(TIME_SCALE 1.000E-12)
(CAP_SCALE 1.000E-12)
(RESISTANCE_SCALE 1.000E+00)
(DEFAULT_INPUT_SLEW 0.000000 0.000000 0.000000 0.000000)
(PROGRAM "novus" "v13.10-dev.festa ((64bit) 04/02/2013 23:24 (Linux 2.6))" "Cadence")
)
(WAVEFORM "Iclk1" 10000.000000 (POSEDGE 0.000000) (NEGEDGE 5000.000000))
(WAVEFORM "Iclk2" 15000.000000 (POSEDGE 0.000000) (NEGEDGE 7500.000000))
(WAVEFORM "Iclk3" 15000.000000 (POSEDGE 0.000000) (NEGEDGE 7500.000000))
(WAVEFORM "Iclk4" 10000.000000 (POSEDGE 0.000000) (NEGEDGE 5000.000000))
(CAUSED_BY "Iclk1" RISE
(NET "in1" 0:200 120:120 * 9200 0:200 120:120 * 9200 D)
)
(CAUSED_BY NULL
(NET "sel" 0:200 120:120 * * 0:200 120:120 * * D)
)
(CAUSED_BY "Iclk4" RISE
(NET "clk1" 0:200 120:120 * * * 120:120 * * D)
)
(CAUSED_BY "Iclk4" FALL
(NET "clk1" * 120:120 * * 5000:5200 120:120 * * D)
)
```

```
(CAUSED_BY "Iclk1" RISE
(NET "clk1" 0:200 120:120 * * * 120:120 * * D)
)
(CAUSED_BY "Iclk1" FALL
(NET "clk1" * 120:120 * * 5000:5200 120:120 * * D)
)
(CAUSED_BY "Iclk1" RISE
(NET "clk1" 0:200 120:120 * * 5000:5200 120:120 * * C)
)
(CAUSED_BY "Iclk4" RISE
(NET "clk1" 0:0 120:120 * * 5000:5000 120:120 * * C)
)
(CAUSED_BY NULL
(NET "clk2" 0:200 120:120 * * 0:200 120:120 * * D)
)
(CAUSED_BY "Iclk1" RISE
(NET "out1" 1200:2600 12.7:12.7 * 1300:7400 1200:2600 10.8:10.8 * 1300:7400 D)
)
(CAUSED_BY "Iclk4" RISE
(NET "out1" 300:800 12.7:12.7 * 400:9200 300:800 10.8:10.8 * 400:9200 D)
```

Related Commands

- read_twf

set_socv_constraint_config

```
set_socv_constraint_config
[-help]
[-hold_folding float]
-libs string
[-nochange_hold_folding float]
[-nochange_setup_folding float]
[-setup_folding float]
```

Enhances the built-in sigma pessimism of check values for setup and hold constraints in the library. The folding factor attribute records the amount of on-chip-variation pessimism added during cell characterization to the nominal setup/hold tables in the libraries. The folding factor attributes are library level attributes.

Notes

- The `set_socv_constraint_config` command does not reset the timing; so it should be used before the `report_timing` command.
- To change the setting later in the flow, you need to use the `update_timing -full` command.
- You cannot use this command, if any of the following global variables are specified:
  `timing_library_setup_sigma_multiplier`
  `timing_library_hold_sigma_multiplier`

Parameters

`-help`	Prints out the command usage.
`-hold_folding` *`float`*	Specifies a number of additional sigma units added on top of the existing nominal hold value. *Min* = -100 *Max* = 100
`-libs` *`string`*	Specifies a list of library names (not library paths) or collection objects.
`-nochange_hold_folding` *`float`*	Specifies a number of additional sigma units added on top of the existing nominal hold value for no-change timing arcs. *Min* = -100 *Max* = 100
`-nochange_setup_folding` *`float`*	Specifies a number of additional sigma units added on top of the existing nominal setup value for no-change timing arcs. *Min* = -100 *Max* = 100
`-setup_folding` *`float`*	Specifies a number of additional sigma units added on top of the existing nominal setup value. *Min* = -100 *Max* = 100

Examples

- The following command is used when an additional margin of 10 sigma units is applied to the nominal hold tables in the library collection test.lib:

```
set_socv_constraint_config -hold_folding 10 -libs [get_libs test.lib]
```

Related Commands

- report_timing
- update_timing

get_socv_rc_variation_factor

```
get_socv_rc_variation_factor
[-help]
-view viewName
{-early | -late}
```

Displays the variation factor for interconnect delays.

Parameters

`-help`	Prints out the command usage.
`-early`	Displays the variation factor for early interconnect delays.
`-late`	Displays the variation factor for late interconnect delays.
`-view viewName`	Displays the SOCV interconnect variation factor for specified view.

Related Commands

- set_socv_rc_variation_factor

create_property_alias

```
create_property_alias
[-help]
-map <alias_propery_name_pairs>
-obj_type {pin port cell net clock lib_cell lib_pin design lib timing_path timing_point
timing_arc path_group lib_timing_arc si_victim si_attacker lib_pg_pin pg_pin pg_net}
```

Creates a property alias for the selected objects.

Parameters

`-help`	Prints out the command usage.
`-map <alias_propery_name_pairs>`	Specifies pair list of alias and property to be associate with that alias, such as, {alias1 prop1} or {{ialias1 prop1}{alias2 prop2}...}.
`-obj_type {pin port cell net clock lib_cell lib_pin design lib timing_path timing_point timing_arc path_group lib_timing_arc si_victim si_attacker lib_pg_pin pg_pin pg_net}`	Specifies the object type for which a property alias needs to be created.

get_clock_to_clock_association

```
get_clock_to_clock_association
[-help]
<clock_list>
[-allow_paths]
[-group_name <group_name>]
[-type {logically_exclusive | physically_exclusive | asynchronous}]
[-view <viewName>]
```

Reports clock to clock associations.

Parameters

`-help`	Prints out the command usage.
`<clock_list>`	Specifies a list of clocks that must contain two entries.
`-allow_paths`	Checks whether the two specified clocks are asynchronous and timing analysis is allowed between these clocks.
`-group_name <group_name>`	Reports clock to clock association for the specified clocks along with the clock group name.
`-type {logically_exclusive physically_exclusive asynchronous}`	Checks clock to clock association for the specified type of clocks. You can use one of the following options: • `logically_exclusive`: Reports clock group with clocks that do not have any functional paths between them, but coexist in the design at the same time. • `physically_exclusive`: Reports clock group with clocks that cannot exist in the design at the same time. • `asynchronous`: Reports clock group with asynchronous clocks. Asynchronous clocks have no specified phase relationship.
`-view <viewName>`	Generates a report for the specified analysis view.

Related Commands

- create_clock
- create_generated_clock
- set_clock_groups

report_lib_arcs

```
report_lib_arcs
[-help]
-arc <string>
[-fall]
[-nldm]
[-quiet]
[-rise]
[-lvf  [-early]]
[-lvf  [-late]]
[-moments  [-mean_shift]]
[-moments  [-std_dev]]
[-moments  [-skewness]]
[-input_net_transition <float> | -constrained_pin_transition <float>]
[-input_net_transition <float> | -related_pin_transition <float>]
[-total_output_net_capacitance <float> | -constrained_pin_transition <float>]
[-total_output_net_capacitance <float> | -related_pin_transition <float>]
[-constrained_pin_transition <float> | -delay ] [-constrained_pin_transition <float> | -
transition ]
[-related_pin_transition <float> | -delay ]
[-related_pin_transition <float> | -transition ]
```

Reports the library arc data. The command also returns the data in a Tcl object, which can be used in scripts.

This command uses a collection of arcs as input, which can be queried using the `get_lib_arcs` command.

The units for the input and output data can be obtained using the `get_time_unit` and `get_capacitance_unit` commands.

Parameters

`-help`	Prints out the command usage.
`-arc <string>`	Specifies the arc(s) for which the report is to be printed.

`-constrained_pin_transition <float>`	Specifies the constrained pin slew (valid for constraint arcs) at which the parameters need to be reported. The slew units must be specified using the `get_time_unit` command.
`-delay`	Specifies if delay is to be reported.
`-early`	Specifies if early LVF data is to be reported. This parameter can be used only in conjunction with the `-lvf`.
`-fall`	Specifies if details of fall tables are to be reported.
`-input_net_transition <float>`	Specifies the slew at which the parameters need to be reported. The slew units must be specified using the `get_time_unit` command.
`-late`	Specifies if late LVF data is to be reported. This parameter can be used only in conjunction with the `-lvf`.
`-lvf`	Specifies if LVF data is to be reported.
`-mean_shift`	Specifies if the mean shift data is to be reported. This parameter can be used only in conjunction with the `-moments`.
`-moments`	Specifies if moments data is to be reported.
`-nldm`	Specifies if NLDM data is to be reported.
`-quiet`	Disables output to be displayed on the screen.
`-related_pin_transition <float>`	Specifies the related pin slew (valid for constraint arcs) at which the parameters need to be reported. The slew units must be specified using the `get_time_unit` command.
`-rise`	Specifies if details of rise tables are to be reported.

`-skewness`	Specifies if skewness data is to be reported. This parameter can be used only in conjunction with the `-moments`.
`-std_dev`	Specifies if standard deviation data is to be reported. This parameter can be used only in conjunction with the `-moments`.
`-total_output_net_capacitance <float>`	Specifies the load at which the parameters need to be reported. The load must be specified using the `get_capacitance_unit` command.
`-transition`	Specifies if the output slew is to be reported.

Examples

- The following command reports the library arc data containing mean shift rise tables:

```
report_lib_arcs -arc [get_lib_arcs -from cell_w/DFF/CK -to cell_w/DFF/Q] -
input_net_transition 0.08 /
       -total_output_net_capacitance 0.055 -mean_shift -rise -delay -moments -nldm

Units of Report:- Time: '1000ps', Capacitance: '1000fF'.
input_net_transition:0.080000, total_output_net_capacitance:0.055000
-------------------------------------------------------------------
| Index | Timing Arc | Value(s) |
| ---------------------------------------------------------------------------|
|       | Cell       | Source/Sink Pin | Timing Sense | Timing Type | cell_rise |
-----------------------------------------------------------------------------
| 1     | cell_w/DFF | CK -> Q         | non_unate    | rising_edge | 0.258444  |
-----------------------------------------------------------------------------
{{cell_rise 0.258444279432} {ocv_mean_shift_cell_rise NA}}
```

- The following command reports the library arc data containing fall tables for moments data:

```
report_lib_arcs -arc [get_lib_arcs -from cell_w/BUF/A -to cell_w/BUF/Y] /
```

```
    -input_net_transition 0.055 -total_output_net_capacitance 0.15 -fall -delay -
moments -nldm

Units of Report:- Time: '1000ps', Capacitance: '1000fF'.
input_net_transition:0.055000, total_output_net_capacitance:0.150000
-------------------------------------------------------------------------------
| Index | Timing Arc                                                  | Value(s)   |
| ------------------------------------------------------------------------------|
|       | Cell       | Source/Sink Pin | Timing Sense   | Timing Type  | cell_fall  |
-------------------------------------------------------------------------------
| 1     | cell_w/BUF | A -> Y          | positive_unate | combinational| 0.129744   |
-------------------------------------------------------------------------------
-------------------
{{cell_fall 0.129} {ocv_std_dev_cell_fall NA} {ocv_mean_shift_cell_fall NA}
{ocv_skewness_cell_fall NA}}
```

- The following command reports the library arc data containing rise tables data with output slew:

```
report_lib_arcs -arc [get_lib_arcs -from cell_w/DFF/CK -to cell_w/DFF/Q] -
input_net_transition 0.08 /
     -total_output_net_capacitance 0.055 -mean_shift -rise -delay -moments -nldm -
transition

Units of Report:- Time: '1000ps', Capacitance: '1000fF'.
input_net_transition:0.080000, total_output_net_capacitance:0.055000
-------------------------------------------------------------------------------
--------------
| Index | Timing Arc
| Value(s)  |
| ------------------------------------------------------------------------------
--------------
|       | Cell       | Source/Sink Pin | Timing Sense | Timing Type | rise_transition
| cell_rise |
-------------------------------------------------------------------------------
--------------
| 1     | cell_w/DFF | CK -> Q         | non_unate    | rising_edge | 0.21692
 | 0.258444  |
-------------------------------------------------------------------------------
--------------
{{cell_rise 0.258} {rise_transition 0.216} {ocv_mean_shift_cell_rise NA}
{ocv_mean_shift_rise_transition NA}}
```

This command also returns the data in a Tcl object that can be parsed and used in scripts. A 3-Dimensional Tcl List will return the following information:

{{TableName Value} {TableName Value}} {{TableName Value} {TableName Value}}

Here, the first list (left column) indicates the Table-Value pair in one arc. The second list indicates the various tables present per arc. And the third list (last column) indicates the list of total arcs in the collection.

- The following command reports the slew and capacitance load:

```
set items [report_lib_arcs -arc [ get_lib_arcs -from sc_cmoslp/AO22X/clk -to
sc_cmoslp/AO22X /out1] /
     -input_net_transition 0.003 -total_output_net_capacitance 0.021]


Units of Report:- Time: '1000ps', Capacitance: '1000fF'.
input_net_transition:0.003, total_output_net_capacitance:0.021
-------------------------------------------------------------------------------------
-----------------------------------------------------
| Index | Timing Arc
                                        | Value(s) |
|------------------------------------------------------------------------------------
-----------------------------------------------------
|       | Cell             | Source/Sink Pin | Timing Sense   | Timing Type   |
rise_transition | fall_transition | cell_rise | cell_fall |
-------------------------------------------------------------------------------------
-----------------------------------------------------
| 1     | sc_cmoslp/AO22X | clk -> out1     | positive_unate | combinational |
0.20**          | 0.30**          | 1.00**    | 1.20**    |
-------------------------------------------------------------------------------------
-----------------------------------------------------
| 2     | sc_cmoslp/AO22X | clk -> out1     | negative_unate | combinational |
0.20**          | 0.30**          | 1.00**    | 1.20**    |
-------------------------------------------------------------------------------------
-----------------------------------------------------

** --> Scalar Value{{cell_rise 0.999} {rise_transition 0.199} {cell_fall 1.199}
{fall_transition 0.299}} {{cell_rise 0.999} {rise_transition 0.199} {cell_fall 1.199}
{fall_transition 0.299}}

tempus 5 > puts $items
{{cell_rise 0.999} {rise_transition 0.199} {cell_fall 1.199} {fall_transition 0.299}}
{{cell_rise 0.999} {rise_transition 0.199} {cell_fall 1.199} {fall_transition 0.299}}
```

Related Commands

- get_lib_arcs
- get_time_unit

- get_capacitance_unit

list_libraries

```
list_libraries
[-help]
[-power_domain <string>]
[-used]
[-used_analysis]
[-view <string>]
```

Lists all the libraries loaded in the session and used in the design.

Parameters

`-help`	Prints out the command usage.
`-power_domain <string>`	Specifies the power domain from which instances are to be selected.
`-used`	Prints out list of only those libraries that are being used.
`-used_analysis`	Prints a list of used libraries in the design.
`-view <string>`	Specifies the view name whose libraries will be listed.

Examples

Here are some examples.

- The following command reports all the libraries loaded in a session:

```
list_libraries

Library Library Library  Group Name Path
------- ------- -------  max fast
/home/fetiming/cte/regressions/data/libs/tsmc18/fast.lib
min slow /home/fetiming/cte/regressions/data/libs/tsmc18/slow.lib
```

Related Commands

- `current_design`

38

Timing Budgeting Commands and Attributes

- budget Category Attributes
- check_partition_sdc
- create_timing_budget
- report_timing_budget
- report_timing_budget_exception
- reset_modified_budget
- set_cycle_budget_ratio
- set_fixed_budget
- set_partition_user_constraints_file
- update_timing_budget
- write_timing_budgets

budget Category Attributes

The root attributes in the `budget` category are:

- budget_abutted
- budget_accumulated
- budget_boundary_condition_template_path
- budget_buffer_delay_adjustment
- budget_buffer_delay_lib_cell
- budget_buffer_delay_selection_effort
- budget_constant_model
- budget_distribute_mmmc
- budget_drive_cell
- budget_fix_top_level_paths
- budget_handle_complex_sdc
- budget_honor_report_timing_format
- budget_ignore_dont_touch
- budget_include_latency
- budget_include_wire_loads_in_lib
- budget_input_load
- budget_input_transition
- budget_justify
- budget_keep_pin_list_for_block_ports
- budget_latency_on_clocks
- budget_local_latency
- budget_local_uncertainty
- budget_make_negative_input_delay_zero
- budget_master_clone

- budget_no_false_paths_for_unconstrained_ports
- budget_no_hold_view
- budget_no_setup_view
- budget_override_net_cap
- budget_pin_load
- budget_report_dir
- budget_report_negative_slack_on_ports
- budget_report_or_update_budget
- budget_snap_feedthru_budget_to
- budget_snap_input_budget_to
- budget_snap_negative_only
- budget_snap_output_budget_to
- budget_stage_based_fanout_driving_factor
- budget_stage_based_partition_multiplier
- budget_stage_based_weight
- budget_top_level
- budget_top_level_delay_per_length
- budget_top_level_min_delay_per_net
- budget_use_boundary_condition
- budget_use_real_cell_for_timing_model
- budget_virtual_opt_engine
- budget_write_constraints_for_clock_output_ports
- budget_write_false_path_for_hold
- budget_write_latency_per_clock

To set the value of any root attribute, use the `set_db` command:

```
set_db attribute_name value
```

To get the current value of any root attribute, use the `get_db` command:

```
get_db attribute_name
```

To get the current value of all attributes in the `budget` category, use the following command:

```
get_db -category budget
```

budget_abutted

```
budget_abutted {true | false}
```

Default: `false`

Data_type: bool, read/write

Specifies that the software abuts the partitions when allocating budgets from positive slack paths. With this attribute, the software assumes that all partitions are abutted and therefore the software does not allocate extra delay at the top-level.

Note: The software automatically handles the abutted partitions (both partial and full) even when you do not specify this attribute.

budget_accumulated

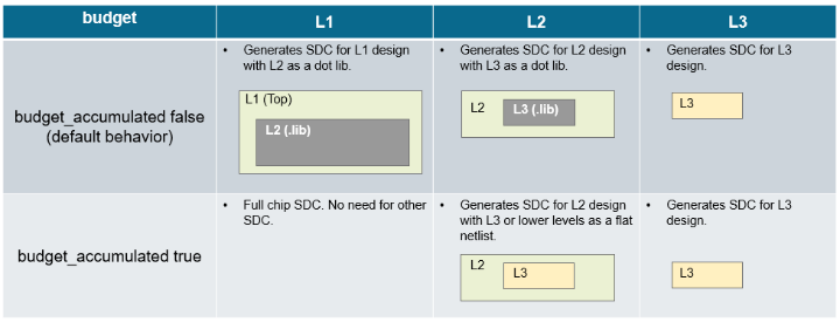
```
budget_accumulated {true | false}
```

Default: `false`

Data_type: bool, read/write

Generates SDC for lower-level hierarchy child partition in the .lib and flat forms (`L3` as shown in the example below). The following example also presents the

matrix to understand the behavior of the `budget_accumulated` attribute at different levels of hierarchy:



With the help of the `budget_accumulated` attribute, SDC supports generation of constraints in the specified child partition through a variable-enabled conditional loop, as shown below:

```
if {[get_db budget_use_accumulated] != 1}{
#Constraints considering child partition in the .lib form
} else {
#Constraints considering child partition in the flat form
}
```

Note:

1. A single `viewDefinition.tcl` or SDC file can support both the .lib or flat flows. This is because there are conditional loops in the `viewDefinition.tcl` and SDC files.
2. Conditional loops are controlled by a variable named `budget_use_accumulated`. The default value of this variable is `0`, which specified that constraints will be processed considering the child partition in the .lib form. You need to run `set_db budget_use_accumulated 1` before loading the design for processing the constraints considering the child partition in the flat form.
3. This attribute does not impact the output of the `viewDefinition.tcl` and SDC files for the bottom-most partitions.

budget_boundary_condition_template_path

`budget_boundary_condition_template_path` *`directory_path`*

Default: `""`

Data_type: string, read/write

Specifies the directory path containing the model files to be used for generating template-based boundary condition through the `budget_use_boundary_condition template` attribute. Hence, the `budget_boundary_condition_template_path` attribute allows you to specify your own template file for generating boundary conditions.

The directory path specified with this attribute should have boundary model files for each view with the file name format as `view_name.model`. If the model files are missing for any view, an error is prompted. If a view is both `setup` and `hold`, the same model will be used.

budget_buffer_delay_adjustment

`budget_buffer_delay_adjustment` *`value`*

Default: `0.0`

Data_type: double, read/write

Specifies the delay for virtual buffer adjustment. If you specify `0`, there will be no adjustment.

If this attribute is not set, timing budgeting performs regular virtual buffering.

budget_buffer_delay_lib_cell

`budget_buffer_delay_lib_cell` *`libname/cellname`*

Default: `none`

Data_type: string, read/write

Specifies the buffer that should be used for virtual buffer adjustment. This is applicable in cases of regular virtual buffering, which can be overridden by the

`budget_buffer_delay_adjustment` attribute.

budget_buffer_delay_selection_effort

`budget_buffer_delay_selection_effort {low|high}`

Default: `low`

Data_type: enum {low high}, read/write

Controls the virtual buffer selection effort for calculating the buffer delay adjustment. The two effort levels are:

- `low` : use buffer selection algorithm based on drive strength
- `high` : use buffer selection algorithm based on actual one buffer delay

budget_constant_model

`budget_constant_model {true | false}`

Default: `true`

Data_type: bool, read/write

Specifies the type of timing model created. The `budget_constant_model` attribute specifies that constant timing models are created.

budget_distribute_mmmc

`budget_distribute_mmmc {true | false}`

Default: `false`

Data_type: bool, read/write

Controls the distribution of MMMC views for budgeting. The distribution could be on the same machine or different machines on a farm/lsf.

budget_drive_cell

`budget_drive_cell {true | false}`

Default: `false`

Data_type: bool, read/write

Writes out the boundary drive information for input pins in the generated SDC file, using the `set_driving_cell` constraint. If not specified, the `set_drive` constraint is written.

This is used to model transition on input port based on load attached to the drive cell.

budget_fix_top_level_paths

`budget_fix_top_level_paths {none | all | negative_only | positive_only}`

Default: `none`

Data_type: enum, read/write

Fixes the top-level timing budget and proportions the remaining timing budget only for the partitions based on the specified value. Use one of the following attribute values:

- `none`: Fixes the default proportionate budgeting.
- `all`: Fixes budgeting only for the partitions.
- `negative_only`: Fixes budgeting only for the partitions with negative slack paths.
- `positive_only`: Fixes budgeting only for the partitions with positive slack paths.

budget_handle_complex_sdc

`budget_handle_complex_sdc {true | false}`

Default: `false`

Data_type: bool, read/write

Specifies to use Common Timing Engine's `write_sdc` for correct push down in case of nested/complex SDC constraints. It can be used when there are regular expressions in SDC constraints.

budget_honor_report_timing_format

`budget_honor_report_timing_format {true | false}`

Default: `false`

Data_type: bool, read/write

Honors the attribute set through `timing_report_fields` by the user in the justification reports, which are generated through the `budget_justify` attribute. This attribute keeps justification reports and columns of `report_timing` in the same format as set by the user.

budget_ignore_dont_touch

`budget_ignore_dont_touch {true | false}`

Default: `false`

Data_type: bool, read/write

Specifies the handling of don't touch (fixed) objects. The attribute allows you to modify these objects also.

budget_include_latency

`budget_include_latency {true | false}`

Default: `true`

Data_type: bool, read/write

Specifies whether the clock latency is included in the `set_input_delay` and `set_output_delay` constraints.

Note: The clock latency is always included if the design has propagated clock(s) irrespective of this attribute.

`set_input_delay` and `set_output_delay` change and are based on actual arrival times.

budget_include_wire_loads_in_lib

`budget_include_wire_loads_in_lib {true | false}`

Default: `false`

Data_type: bool, read/write

Specifies to include the wire loads in library model pin capacitance to handle high interface net loads. It provides extra information in.*lib* file for more accurate modelling.

budget_input_load

`budget_input_load {true | false}`

Default: `false`

Data_type: bool, read/write

Specifies whether the `set_load` constraints have to be written for input pins in the generated SDC file. With this option, load constraint on ports is specified.

budget_input_transition

`budget_input_transition {true | false}`

Default: `false`

Data_type: bool, read/write

Writes out input transition time in the budgeting output files for input pins in the generated SDC file, using the `set_input_transition` constraint. If not specified,

the `set_drive` constraint is written. You must enter the transition value directly.

budget_justify

`budget_justify {delay | exception | lib_arc | boundary_condition | all}`

Default: `""`

Data_type: enum*, read/write

Controls the type of justification and also decides whether or not to generate the justification reports. You may use one or more of the following values with this attribute:

- `delay`: Budget justification files for all the partition IO ports are generated.
- `exception`: Budget justification files for all the partition IO ports are generated with exception data.
- `lib_arc`: Justification files for libs are generated.
- `boundary_condition`: Justification files for the boundary conditions used in the constraint files are generated.
- `all`: All the justification files are generated.

Note: When this attribute is used without any value, no justification files are generated. Only the summary is generated.

budget_keep_pin_list_for_block_ports

`budget_keep_pin_list_for_block_ports {true | false}`

Default: `true`

Data_type: bool, read/write

Specifies to preserve all fan-in and fan-out pin lists for block ports.

The default value is `true`. When set to `true`, the run-time will be the fastest but involves high memory usage. The value of `false` means that the peak memory consumed will decrease but with an increase in run-time.

You can use this attribute to control memory consumption and run-time for budgeting processes. It does not affect other functions when specified.

budget_latency_on_clocks

`budget_latency_on_clocks {true | false}`

Default: `false`

Data_type: bool, read/write

Writes the `set_clock_latency` constraint on the clocks instead of ports in the generated SDC files. Use this attribute when you have multiple clocks on the same port.

Default: Writes the `set_clock_latency` constraint on the ports.

budget_local_latency

`budget_local_latency {true | false}`

Default: `false`

Data_type: bool, read/write

Adjusts the `set_clock_latency` constraint information that you specify using the `set_partition_user_constraints_file` command to the constraints file generated after budgeting.

When you specify this attribute, the software checks whether the `set_clock_latency` constraint specifies a clock or a port.

If the constraint specifies a clock and the clock exists in the partition, then the constraint is added to the constraints file generated after budgeting. If the clock does not exist in the partition, the software does not write out the constraint.

If the constraint specifies a port, the constraint is added to the constraints file generated after budgeting.

The source latency is calculated as follows:

source latency = top-level source latency + top-level network latency - local network latency

budget_local_uncertainty

`budget_local_uncertainty {true | false}`

Default: `false`

Data_type: bool, read/write

Adds the `set_clock_uncertainty` constraint information that you specify using the `set_partition_user_constraints_file` command to the constraints file generated after budgeting.

When you use this attribute, the constraint file generated after budgeting has input and output delays using virtual clocks instead of real clocks in the SDC file. The software uses the virtual clocks to differentiate the clock uncertainty for external paths (paths that are coming from or going to the top-level) and internal paths of the partition.

budget_make_negative_input_delay_zero

`budget_make_negative_input_delay_zero {true | false}`

Default: `false`

Data_type: bool, read/write

Specifies to adjust negative set input delay to 0.

budget_master_clone

`budget_master_clone {master_only | merged | unique_view_per_hinst}`

Default: `master_only`

Data_type: enum {master_only merged unique_view_per_hinst}, read/write

Specifies how the constraints are generated for master and clone designs.

budget_no_false_paths_for_unconstrained_ports

`budget_no_false_paths_for_unconstrained_ports {true | false}`

Default: `false`

Data_type: bool, read/write

Specifies not to create false paths for unconstrained ports.

budget_no_hold_view

`budget_no_hold_view {true | false}`

Default: `false`

Data_type: bool, read/write

Specifies to derive budgets only for setup views. Dummy files are created for the hold view as budgeting is not done for them.

budget_no_setup_view

`budget_no_setup_view {true | false}`

Default: `false`

Data_type: bool, read/write

Specifies to derive budgets only for hold views. Dummy files are created for the setup view as budgeting is not done for them.

budget_override_net_cap

`budget_override_net_cap` *value*

Default: `100ff`

Data_type: double, read/write

Specifies the lump capacitance value. The software uses this value when you use the `preliminary` attribute in the `create_timing_budget` command.

budget_pin_load

`budget_pin_load {true | false}`

Default: `false`

Data_type: bool, read/write

Writes out the `set_load` with `pin_load` information for output pins in the generated SDC file.

budget_report_dir

`budget_report_dir` *`directory_name`*

Default: `budget_justify`

Data_type: string, read/write

Directs output of budget justification to the specified directory.

budget_report_negative_slack_on_ports

`budget_report_negative_slack_on_ports` *value*

Default: `0`

Data_type: double, read/write

Reports if partition ports have a slack less than the specified value and saves this information in the.warn file. The slack is calculated based on data arrival time and clock time at data sampling points. The report is saved to the.warn file.

budget_report_or_update_budget

`budget_report_or_update_budget {true | false}`

Default: `false`

Data_type: bool, read/write

Specifies to enable the collection of data required to use the `update_timing_budget` command. You must specify this attribute before budgeting to run `update_timing_budget` later in the flow.

If this attribute is not specified, the `update_timing_budget` command does not produce any results.

budget_snap_feedthru_budget_to

`budget_snap_feedthru_budget_to` *`value`*

Default: `0`

Data_type: double, read/write

Specifies the minimum delay value (in picoseconds) for the path from partition input port to partition output port. If, during timing analysis, the path segment delay used to generate budgeting constraints for the partition ports falls below the specified threshold value, the delay segment is snapped to the specified value and is considered as a fixed delay during budget allocation for the path.

The.`warn` file does not issue a warning for feedthrough segments. By default, the software distributes a delay equivalent of two medium-sized buffer delays among the feedthrough segments. As a result, it is unlikely that the feedthrough segment delay will fall below a reasonable value.

budget_snap_input_budget_to

`budget_snap_input_budget_to` *`value`*

Default: `0`

Data_type: double, read/write

Specifies the minimum delay (in picoseconds) for the path from the partition input port to the internal register. If, during timing analysis, the path segment delay falls below the specified threshold value, the delay segment is snapped to the specified value and is considered as a fixed delay during budget allocation for

the path.

Note: During timing analysis, the path segment delay could be greater then the specified value; however, after budget allocations, the allocated delay for the partition can still become smaller than the specified threshold value. The software checks the delays allocated to partitions for input and output paths and creates appropriate warning messages in the partition.`warn` file.

budget_snap_negative_only

```
budget_snap_negative_only {true | false}
```

Default: `false`

Data_type: bool, read/write

Considers only the negative slack paths when used with `budget_snap_input_budget_to` and `budget_snap_output_budget_to`.

budget_snap_output_budget_to

```
budget_snap_output_budget_to value
```

Default: `0`

Data_type: double, read/write

Specifies the min delay (in picoseconds) for the path from the internal register to the partition output port. If, during timing analysis, the path segment falls below the specified threshold value, then the delay segment is snapped to the specified value and is considered as a fixed delay during budget allocation for the path.

budget_stage_based_fanout_driving_factor

```
budget_stage_based_fanout_driving_factor value
```

Default: `-1`

Data_type: int, read/write

Specifies the fanout count of the pin, according to which the buffer stages are added in the timing graph. This attribute provides the basis for the software to calculate the number of fanouts and buffer stages to be virtually added for each stage according to the following table:

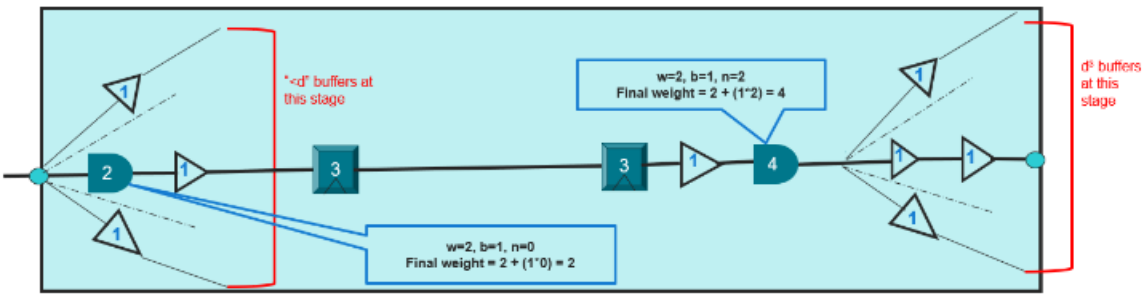
Fanout Range	Number of Buffer Stage Added (n)
1 to d	0
(d+1) to d2	1
(d2+1) to d3	2
(d3+1) to d4	3
..	..

If the weight of the original cell is w, weight of the buffer is b, and the number of buffer stages is n, then:

Final weight of cell = w + (b*n)

See the example below for understanding the functioning of the `budget_stage_based_fanout_driving_factor` attribute.

```
set_db budget_stage_based_fanout_driving_factor {3 2 1}
```



budget_stage_based_partition_multiplier

```
budget_stage_based_partition_multiplier {{HInst1 m1} {HInst2 m2}..}
```

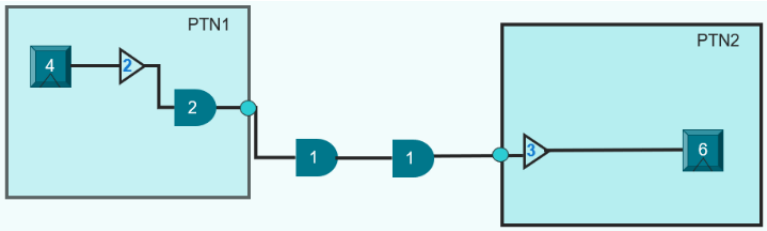
Default: `""`

Data_type: string, read/write

Specifies the weight multiplication factor for the partition hinsts. Those hInsts, which are not explicitly mentioned, will have the default weights. If the weights of hinsts have been specified using the `budget_stage_based_weight` attribute, those are used with their default weights. Else, the default weights (1 for sequential cell, 1 for combinational cell, and 0 for buffer/inverter) are used. In case of nested partitions, if the child partition multiplier is not specified explicitly, then it will have its parent's multiplier. If parent and child partitions have separate multipliers, then they will be honored.

See the example below for understanding the usage of this attribute:

```
set_db budget_stage_based_weight {2 1 1}
set_db budget_stage_based_partition_multiplier {{PTN1 2} {PTN2 3}}
Cycle time = T
Budget for PTN1= 8/19 T
Budget for PTN2= 9/19 T
Budget for top = 2/19 T
```



budget_stage_based_weight

```
budget_stage_based_weight {S | C | B}
```

Default: `{1 1 0}`

Data_type: string, read/write

Specifies default weights of the sequential, combinational, and buffer/inverter cells for the stage-based timing budgeting. Following are the default weights of these cells:

- Sequential: 1
- Combinational: 1
- Buffer/inverter: 0

budget_top_level

```
budget_top_level minimumValue
```

Default: `-1` (indicates regular slack or reset of the top level)

Data_type: double, read/write

Specifies the minimum percentage of total available budget set aside for the top level. This attribute can take any value between 0 - 1.

- `0` - indicates no slack to top (100% to block).
- `1` - maximum value that indicates full slack to top (0 % block)

budget_top_level_delay_per_length

```
budget_top_level_delay_per_length value
```

Default: `180ps/mm`

Data_type: int, read/write

Specifies the top-level estimated delay (in picoseconds) per millimeter length. The software uses this value when you use the `preliminary` attribute in the `create_timing_budget` command.

For more information on the use of the budget_top_level_delay_per_length attribute, see "Deriving Preliminary Budgets in Early Design Phase" in the Timing Budgeting chapter of the *Innovus User Guide* document.

budget_top_level_min_delay_per_net

```
budget_top_level_min_delay_per_net value
```

Default: 100ps/mm

Data_type: int, read/write

Specifies the top-level minimum delay-per-net value in picoseconds/millimeter units. The software uses this value when you use the preliminary attribute in the create_timing_budget command.

For more information on the use of the budget_top_level_min_delay_per_net attribute, see "Deriving Preliminary Budgets in Early Design Phase" in the Timing Budgeting chapter of the *Innovus User Guide* document.

budget_use_boundary_condition

```
budget_use_boundary_condition {optimized | template | actual | empty}
```

Default: `template`

Data_type: enum {optimized template actual empty}, read/write

Specifies the criteria for choosing boundary conditions. Following are the usages of the attribute values:

- `optimized`: The values of boundary conditions are fetched from the engine used to build timing graph. If the engine does not provide any values, than the template is used automatically and a warning is displayed indicating that no optimized boundary conditions are available, and the template is used automatically.
- `template`: The values of boundary conditions are derived automatically and used through the default template.
  Note: The default templates are saved as `budget_model/view_name.model`.
- `actual`: Uses the actual netlist to build the timing graph.
- `empty`: No boundary condition is dumped.

budget_use_real_cell_for_timing_model

```
budget_use_real_cell_for_timing_model {true | false}
```

Default: `false`

Data_type: bool, read/write

Use actual gate connected to the partition port to write out the model timing arc, else medium size buffer is used.

budget_virtual_opt_engine

```
budget_virtual_opt_engine {none | early_timing_engine | proto}
```

Default: `none`

Data_type: enum {none opt early_timing_engine proto}, read/write

Specifies the virtual optimization engine that was used to generate the timing graph. User will have to provide a timing graph generated without using `create_timing_budget` and must set the value of the `budget_virtual_opt_engine` attribute same as that of the command used to generate the timing graph. Following are the `enum` values of this attribute:

- `none`: Specifies that the timing graph is generated through `report_timing`.
- `early_timing_engine`: Specifies that the timing graph is generated by `opt_virtual -early_timing_engine`.
- `proto`: Specifies that the timing graph is generated by prototype timing.

Default: `none`

Note: User can use ART (`create_active_logic_view -type timing_budget`) before generating the timing graph to reduce turnaround time and memory during budgeting flow.

budget_write_constraints_for_clock_output_ports

```
budget_write_constraints_for_clock_output_ports {true | false}
```

Default: `false`

Data_type: bool, read/write

Specifies to write extra constraints on the output port of block SDC. When this attribute is specified, the following extra constraints are written out:

- set_clock_sense - this constraint is dumped if the clock is used as data going forward after partition
- create_generated_clock - this constraint is dumped if the clock is used as clock going forward after partition

If the clock is used as both data and clock going forward, the attribute does not write out these extra constraints.

budget_write_false_path_for_hold

```
budget_write_false_path_for_hold {true | false}
```

Default: `false`

Data_type: bool, read/write

Writes false_path -hold constraints for all I/Os when deriving timing budgets in setup mode.

For inputs, the saveTimingBudget command writes the following constraint:

set_false_path -from port -hold

For outputs, the saveTimingBudget command writes the following constraint:

set_false_path -to port -hold

budget_write_latency_per_clock

```
budget_write_latency_per_clock {true | false}
```

Default: `false`

Data_type: bool, read/write

Specifies to write latency per clock instead of per port. When you set this variable to true, set_clock_latency is written per clock if multiple clocks enter the partition through the same port.

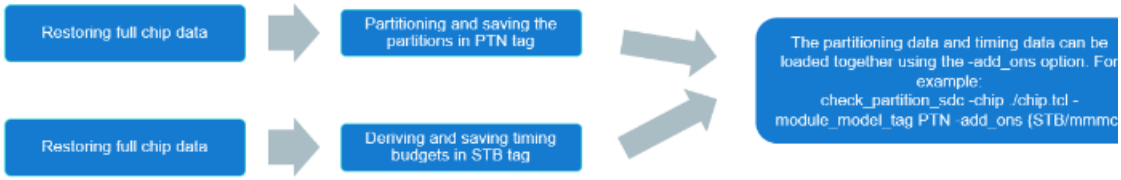
check_partition_sdc

```
check_partition_sdc
[-help]
{[-chip string -partition_dir string] | [-chip string -module_model_tag string [-add_ons string] [-accumulated] [-overwrite_tag]] | [-
compare_only]}
[-cells string]
[-master_only]
[-no_top]
[-out_dir string]
[-validate_boundary_condition]
```

Independently verifies the budgeted block SDC with the chip SDC to reduce the run-time budgeting.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `check_partition_sdc` parameter. For a detailed description of the command and all its parameters, use the man command: `man check_partition_sdc`
`-accumulated`	Runs comparison for the accumulated budgeting mode. *Data_type*: bool, optional

`-add_ons` *string*	Specifies the supplement model that can be loaded additionally with an existing model or a design. It can be used with the `-module_model_tag` parameter. This parameter can only be used in the Integrated Hierarchical Database (iHDB) flow. This parameter is very useful for the flows where partitioning and timing budgeting are done in parallel, as presented below:  *Data_type*: string, optional
`-cells` *string*	Specifies the list of cell names for which the SDC comparison is to be done. *Default:* `all cells` *Data_type*: string, optional
`-chip` *string*	Specifies the TCL file, which loads the chip database. *Data_type*: string, required
`-compare_only`	Compares the dumped data with the previous timing budgeting validation run. *Data_type*: bool, optional
`-master_only`	Dumps and compares constraints for checking only the master partitions for the master-clone designs. In case this parameter not used, the tool dumps and compares constraints taking clones also in consideration. *Data_type*: bool, optional

`-module_model_tag` *string*	Specifies the module model tag for the cell on which the SDC comparison is to be done.
	For example, the following commands validate the timing budgeting results for the specified module model:
	`set_module_model -default_dir ./DATA2`
	`check_partition_sdc -chip scripts/chip.tcl -cells $block -module_model_tag init -no_top`
	Note: This parameter is a part of the iHDB flow. Before using this parameter, you must set the default central directory name (`set_module_model -default_dir` *default_model_data_directory*) where the module models data will be saved/loaded.
	Data_type: string, optional

`-no_top`	Disables checking of the TOP cell. *Data_type*: bool, optional
`-overwrite_tag`	Regenerates design for the accumulated Budgeting mode. *Data_type*: bool, optional
`-out_dir` *string*	Specifies the directory, which contains the output in the form of validation comparison reports and log files. *Default*: `./cps` *Data_type*: string, optional
`-partition_dir` *string*	Specifies the directory, which contains the partition database. *Data_type*: string, required

`-validate_boundary_condition`	Compares the slew values at the chip versus block level. Slew values are generated using the `report_timing` command, and the comparison is done at immediate pins to the block boundary.
	This parameter can be used in the same session in which the default `check_partition_sdc` check is used. However, it should not run in parallel with the default `check_partition_sdc` check since writing of setup file may have a conflict.
	The `-validate_boundary_condition` parameter can validate only the boundary conditions for the partition-block constraints generated for `budget_use_boundary_condition actual` and build the timing graph from the actual netlist. This parameter dumps an output file in the .csv format.
	⚠ The `-validate_boundary_condition` parameter cannot support the `budget_use_boundary_condition {optimized | template}` values. It also gives an error, if the timing graph is not built.
	Following is a sample output of using the `-validate_boundary_condition` parameter:
	`+---------------------------+-----------+------------+----------+`
	`|  Pin  | Chip Slew | Block Slew | Diff (%) |`
	`+---------------------------+-----------+------------+----------+`
	`| slave_haddr_o[0]  |  0.620 |  0.613 |  1.13 |`
	`| rxdata1[6]  |  0.266 |  0.269 |  1.13 |`
	`| rxdata1[7]  |  0.270 |  0.273 |  1.11 |`
	`| master_hready_o  |  0.364 |  0.368 |  1.09 |`
	`| SPCASCAN_N523  |  0.734 |  0.742 |  1.09 |`
	`| rxdata2[1]  |  0.196 |  0.198 |  1.02 |`
	`| slave_hrdata_i[0]  |  0.098 |  0.097 |  0.11 |`
	`+---------------------------+-----------+------------+----------+`
	Data_type: bool, optional

Examples

- The following command runs validation using the chip database `chip.tcl` and post-partition saved along with post-budgeted database in `ptn` tag:
  ```
  check_partition_sdc -chip ./chip.tcl -module_model_tag ptn
  ```
- The following command runs validation using the chip database `chip.tcl`, the post-partition database in the `ptn` tag, and post-budgeted database in the `ptn` tag, which are saved in parallel:
  ```
  check_partition_sdc -chip ./chip.tcl -module_model_tag ptn -add_ons {mmmc}
  ```
- The following command runs validation using the chip database `chip.tcl`, the post-partition database in the `ptn` tag, and post-budgeted database in the `stb` tag:
  ```
  check_partition_sdc -chip ./chip.tcl -module_model_tag ptn -add_ons {STB/mmmc}
  ```
- The following command runs validation using the chip database `chip.tcl` and the partition database in `./PTNS`:
  ```
  check_partition_sdc -chip ./chip.tcl -partition_dir ./PTNS
  ```
- The following command runs validation using the chip database `chip.tcl` and the partition database in `./PTNS` for cells P1 and P2:
  ```
  check_partition_sdc -chip ./chip.tcl -partition_dir ./PTNS -cells {P1 P2}
  ```
- The following command runs validation using the chip database `chip.tcl`, the partition database in `./PTNS`, and the dumping reports and log in `./CHK`:
  ```
  check_partition_sdc -chip ./chip.tcl -partition_dir ./PTNS -out_dir ./CHK
  ```
- The following command runs validation using the chip database `chip.tcl`, and partition database in `./PTNS`, and only checks the partitions data (no portioned top checking with chip):
  ```
  check_partition_sdc -chip ./chip.tcl -partition_dir ./PTNS -no_top
  ```

Related Information

The Timing Budgeting chapter of the *Innovus User Guide* document.

create_timing_budget

```
create_timing_budget
[-help]
[-parameterized]
[-cycle_ratio | -stage_based | -timing_core | -preliminary | -push_down_sdc]
[-all_ilm_hinsts | -hinsts instanceName | -partitions partitionName]
[-verbose]
[-views string]
```

Generates timing budgets for partitions for standalone implementation. This command can also generate timing budgets on an instance basis instead of partitions. The generated timing constraint files allow you to do standalone block implementation. In case of partition based budgeting, it also generates timing models that will allow you to do standalone top level implementation.

Note: You must free the timing graph to use multiple `create_timing_budget` commands with different options.

The Innovus operates in multi-mode multi-corner (MMMC) mode, and generates `view_files` automatically for partition implementation and top-level chip assembly. Note that the `create_timing_budget` command does not generate a timing graph, if it already exists. Additionally, user will have to provide a timing graph generated without using `create_timing_budget` and must set the value of the `budget_virtual_opt_engine` attribute same as that of the command used to generate the timing graph.

When the `create_timing_budget` command is run, all the public budget category attributes with their non-default values are displayed before the intended output of the `create_timing_budget` command, in an information message. This information message follows the format shown below:

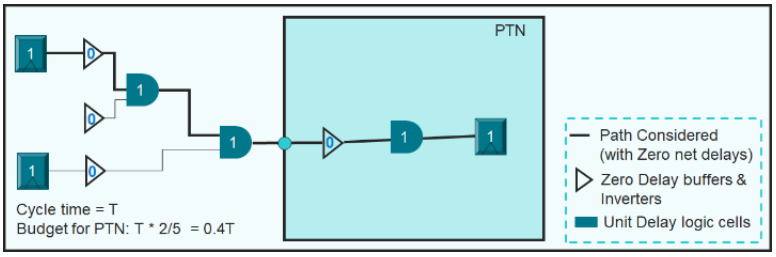
```
This command "create_timing_budget" required an extra checkout of license invs_hier. Additional license(s) checked out: 1 'Innovus_Hier_Opt'
license(s)
**INFO: User settings:
budget_abutted true
budget_fix_top_level_paths all
budget_master_clone merged
budget_no_hold_view true
.
.
```

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `create_timing_budget` parameter. For a detailed description of the command and all its parameters, use the man command: `man create_timing_budget`
`-all_ilm_hinsts`	Derives timing budgets only for all the ILM-based partitions. When this parameter is used, the software automatically derives the list of ILM instances and budgets for them. **Note:** The `-all_ilm_hinsts` parameter internally calls `-inst` based budgeting. So, saving budgets is only supported through the `write_timing_budgets` command. Therefore, it is necessary to run `write_timing_budgets` after `create_timing_budget -all_ilm_hinsts`, else `write_partitions` will not save the budgeting information of ILM instances. *Data_type*: bool, optional

`-cycle_ratio`	Honors the `set_cycle_budget_ratio` constraints and enables usage of clock cycle time based budgeting. If no set_cycle_budget_ratio constraints are applied:
	• The top and block default ratio is 20% and 80% of the clock respectively.
	• The partition for start and end points splits the block ratio in 50% each.
	• Equivalent default constraint is as follows:
	`set_cycle_budget_ratio -from_ptn {* .4} -to_ptn {* .4}`
	`set_cycle_budget_ratio -from_ptn {* .8} -to_top`
	`set_cycle_budget_ratio -from_top -to_ptn {* .8}`
	Note: Do not use the `set_fixed_budget` command with this parameter.
	Data_type: bool, optional

`-hinsts instanceName…`	

	Performs budgeting for the hierarchical instance(s). This instance could be a black box. You can derive timing budgets for black boxes and other instances in the same run. Specify the `-merge_clones` parameter in addition to the `-hinsts` parameter to obtain worst-case budgets based on worst-case data per-pin for specified instances in a master/clone set. If you specify instances from more than one master/clone set, the tool derives separate worst-case timing budgets for the specified instances in each set. *Default:* If you do not specify this parameter, the software derives budgets for all the partitions. *Data_type*: string, optional
`-parameterized`	Generates constraints as a function of the clock period. When you set this parameter, the software generates timing constraints, such as `set_input_delay` and `set_output_delay` constraints as a function of the clock period in the top-level and block-level SDCs. *Data_type*: bool, optional
`-preliminary`	Derives timing budgets in the early stages of the design to get a preliminary estimate of the budgets. When you use this parameter, the software uses the net delays and net loads that you specify using the `setBudgetingMode` command. The software does not use virtual optimization for calculating budgets. The software distributes the positive slack equally between source block, destination block, and top-level. For more information, see the "Deriving Preliminary Budgets in Early Design Phase" section in the Timing Budgeting chapter of the *Innovus User Guide* document. *Data_type*: bool, optional
`-partitions` *partitionName*	
	Performs budgeting for the specified partition(s). Specify the `-merge_clones` parameter in addition to the `-partitions` parameter to obtain worst-case budgets based on worst-case data per-pin for all partitions in the master/clone set containing the partition. You cannot specify `-partitions` if you use `-inst`. *Default:* Performs budgeting on all partitions. *Data_type*: string, optional
`-push_down_sdc`	Helps timing budgeting in creating only pushdown constraints for block partitions. The constraints applied to only those paths, which are completely inside a partition, are known as pushdown constraints. By considering only the pushdown constraints, this parameter ignores the constraints applied on partition interface paths from full chip SDC. **Note:** In the early design phase, the reg-to-IO reg-to-IO and IO-to-reg paths can be ignored using this parameter. In such a case, the timing constraints for only reg-to-reg paths are generated. *Data_type*: bool, optional
`-stage_based`	Creates a quick stage-based timing graph to be used for timing budgeting. Timing budgeting with the stage-based timing graphs is used in very early stages of the design, before applying optimization for quick budgets. This method is more accurate than the cycle-based timing budgeting. When this parameter is used, any existing timing graph in the design is overwritten by the generated special quick stage-based timing graph. The following example presents the scenarios in which this parameter can be used: **reg2reg path**  *Data_type*: bool, optional

`-timing_core`	Generates timing constraints for the partition core (internal register-to-register path only). The following are included in the constraints file: • Clock definitions • Core timing constraints such as `set_multicycle_path`, `set_false_path, and set_max_delay`. These constraints are pushed down from the top-level constraints. *Default:* Generates timing budget files for each partition from the imported top-level timing constraint file. **Note:** The software does not generate budgets for the interface ports of the partition. Therefore the `set_input_delay` or `set_output_delay` values are not generated for the partition ports. *Data_type*: bool, optional
`-verbose`	Enables various verbose controls. *Data_type*: string, optional
`-views` *string*	Derives budget for multiple views in parallel for reducing the runtime. *Data_type*: string, optional

Examples

- For MMMC examples, see the Timing Budgeting chapter of the *Innovus User Guide* document.
- The following command creates a budget for the black box BBox:

```
create_timing_budget -hinsts BBox
write_timing_budgets -hinsts BBox
```

Related Information

The following sections in the Timing Budgeting chapter of the *Innovus User Guide* document:

- Deriving Timing Budgets
  - Budgeting Using Text Commands
  - Top-Level Budgets Derived by Using Active Logic View
  - Deriving Preliminary Budgets in Early Design Phase
- Budgeting Output Files for MMMC Designs
- Analyzing Timing Budgets - Resolving Conflicts with Path-Based Exceptions
- Customizing Budget Generation
- Verifying Timing Budgets
- Reading the Justify Budget Report
- Warning Report

report_timing_budget

```
report_timing_budget
{-pins pinList}
[-out_file fileName]
[-summary]
{partitionName | instanceName}
[-view viewName]
[-help]
```

Queries and verifies the timing budgets generated by Innovus. You can also query budgets selectively for some/all partitions or instances.

The `report_timing_budget` command honors the `timing_report_fields` attribute, which allows you to customize the timing report according to the specified columns. You must set this global before specifying `report_timing_budget` or `create_timing_budget -justify` to get the report in the desired format.

Note: In case of combined budgeting, the `report_timing_budget` command will justify all the input or output delays attached to the port of a partition.

The `report_timing_budget` command only supports the `-summary` parameter in MMMC mode (the default mode). The report will contain only the headers. To print paths, use `create_timing_budget -justify`.

Parameters

`-help`	Prints a brief description that includes the type and default information for each `report_timing_budget` parameter. For a detailed description of the command and all its parameters, use the man command: `man report_timing_budget`
instanceName	Specifies the name of the instance to analyze. *Data_type*: string, required
-out_file fileName	
	Specifies the output file name that contains timing budget data for a pin. *Data_type*: string, optional
partitionName	
	Specifies the name of the partition to analyze. *Data_type*: string, required
-pins pinList	
	Specifies the hierarchical partition pins to be analyzed. You can use `*` to specify all pins. *Data_type*: string, required
`-summary`	Displays the style summary report containing the header section of the full report to show budget calculation only. Use this parameter after using write_timing_budgets/write_partitions. *Data_type*: bool, optional
`-view` *viewName*	Prints a report for the specified view. If you derived the timing budget in MMMC mode (default), this parameter is required to obtain a report. *Data_type*: string, optional

Examples

- For examples of budgeting for MMMC designs, see the Timing Budgeting chapter of the *Innovus User Guide*.
- The following command prints a report for view `VIEW1`

  ```
  report_timing_budget -view VIEW1
  ```

The following example does not apply in MMMC mode, it is shown only for backward compatibility.

- The following commands outputs the timing budget data for a list of pins in `I_SUB`:

```
report_timing_budget -pins in I_SUB
report_timing_budget -pins {in in1 out out2} I_SUB.
```

Related Information

The following sections in the Timing Budgeting chapter of the *Innovus User Guide*:

- Deriving Timing Budgets
  - Budgeting Using Text Commands
  - Top-Level Budgets Derived by Using Active Logic View
  - Deriving Preliminary Budgets in Early Design Phase
- Budgeting Output Files for MMMC Designs
- Analyzing Timing Budgets - Resolving Conflicts with Path-Based Exceptions
- Customizing Budget Generation
- Verifying Timing Budgets
- Reading the Justify Budget Report

report_timing_budget_exception

```
 report_timing_budget_exception
{-pins pinList}
[-out_file fileName]
{partitionName | instanceName}
-view viewName
[-help]
```

Provides a debugging or justification mechanism of exceptions on ports. This command is used to debug exception constraints and justify all exceptions on pins specified by `-pins`.
To justify exceptions, you must use the `budget_justify exception` attribute to save the justify exception data during the `create_timing_budget` process.

Parameters

`-help`	Outputs the command usage.
*partitionName	instanceName*	
	Specifies the name of the partition or instance to analyze. *Data_type*: string, required
`-out_file` *fileName*	
	Specifies the output file name that contains the justify exception data for the instances or partitions specified by *instanceName* or *partitionName*. If you do not specify an output file using the `-out_file` parameter, the exception data will be displayed on the screen. *Data_type*: string, optional
`-pins` *pinList*	
	Specifies the hierarchical partition pins to be analyzed. You can use * to specify all pins. *Data_type*: string, required
`-view` *viewName*	
	Prints a report for the specified view. *Data_type*: string, required

Related Information

The following sections in the Timing Budgeting chapter of the *Innovus User Guide* document:

- Deriving Timing Budgets
  - Budgeting Using Text Commands
  - Top-Level Budgets Derived by Using Active Logic View
  - Deriving Preliminary Budgets in Early Design Phase
- Budgeting Output Files for MMMC Designs
- Analyzing Timing Budgets
  - Resolving Conflicts with Path-Based Exceptions
  - Customizing Budget Generation
- Verifying Timing Budgets
- Reading the Justify Budget Report

reset_modified_budget

```
reset_modified_budget
{-partition partitionName | -inst instanceName}
-pin pinName
[-setup | -hold]
[-view viewName]
[-help]
```

Resets the budget that was modified using the `update_timing_budget` command. `reset_modified_budget` allows you to revert to the original generated budget if the modified budget is not yielding the correct timing for a block, or rectify an incorrectly modified budget.

Parameters

`-help`	Outputs the command usage.
`-pin pinName`	Specifies the name of the pin for which manual budgets are to be reset. You can use `*` to specify all pins. *Data_type*: string, required
`-hold`	Specifies to reset the hold information. *Data_type*: bool, optional
`-inst instanceName`	Specifies the name of the hierarchical instance for which manual budgets are to be reset. *Data_type*: string, required
`-partition partitionName`	Specifies the name of the partition for which manual budgets are to be reset. *Data_type*: string, required
`-setup`	Specifies to reset the setup information. *Data_type*: bool, optional
`-view viewName`	Resets the budget for the specified view. *Data_type*: string, optional

Example

The following command resets the modified budget data for the instance `inst1`:

```
reset_modified_budget -inst inst1
```

Related Information

The 'Modifying Budgets' section in the Timing Budgeting chapter of the *Innovus User Guide* document.

set_cycle_budget_ratio

```
set_cycle_budget_ratio
[-help]
{[[[{-from_top | -from_partition {hinst ratio}} {-to_top | -to_partition {hinst ratio}}] ] [-clock string]] | [-reset ]}
```

Sets the budgeting constraint based on the clock cycle, which ensures that budgeting is done in minimum time and memory by deriving partition block delays from the total clock cycle time. This command supports:

- Different top-level ratios based on critical partition to partition connections
- Different ratios between the partitions with different sources and sinks
- Clock-based segment control

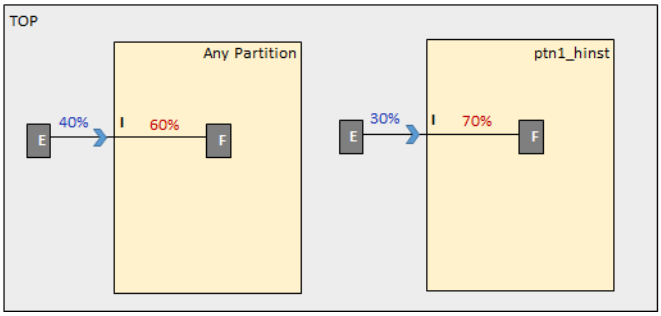
Note: This command (cycle-based budget) does not support nested partition.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `set_cycle_budget_ratio` parameter. For a detailed description of the command and all its parameters, use the man command: `man set_cycle_budget_ratio`
`-clock` *string*	Presents the clock name for which the ratio is to be applied. If specified, the cycle time ratio is applied to the paths only from the specific clock. If not specified, all clocks path are impacted with same percentage of cycle time. *Data_type*: string, optional
`-from_partition {hinst ratio}`	Specifies that the start of a timing path segment is inside the mentioned hierarchical instance, which should get the mentioned cycle ratio for budgeting. *Data_type*: string, required
`-from_top`	Specifies that the start of a timing path segment is at the top level. *Data_type*: bool, required
`-reset`	Resets all the previously declared ratio budgets/constraints. *Data_type*: bool, optional
`-to_partition {hinst ratio}`	Specifies that the end of a timing path segment is inside the mentioned hierarchical instance, which should get the mentioned cycle ratio for budgeting. *Data_type*: string, required
`-to_top`	Specifies that the end of a timing path segment is at the top level. *Data_type*: bool, required

Examples

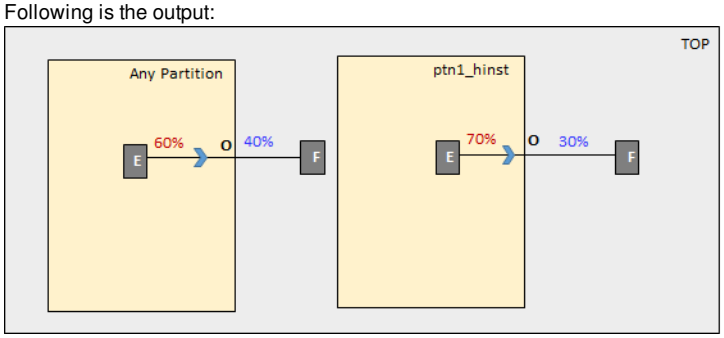
- The following commands present an example of the TOP to PTN path:
  ```
  set_cycle_budget_ratio -from_top -to_partition {* 0.6}
  set_cycle_budget_ratio -from_top -to_partition {ptn1_hinst 0.7}
  ```
  Note: * matches with any partition hinst. However, it does not matches with TOP.
  Following is the output:



- The following commands present an example of the PTN to TOP path:

```
set_cycle_budget_ratio -from_partition {* 0.6} -to_top
set_cycle_budget_ratio -from_partition {ptn1_hinst 0.7} -to_top
```

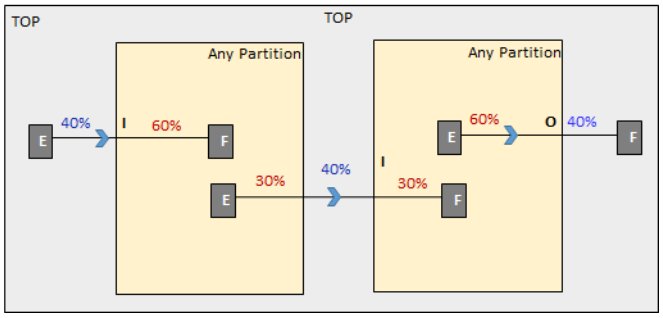
Note: * matches with any partition hinst. However, it does not matches with TOP.
Following is the output:



- The following commands present an example of reserving TOP percentage:

```
set_cycle_budget_ratio -from_partition {* 0.6} -to_top
set_cycle_budget_ratio -from_top -to_partition {* 0.6}
set_cycle_budget_ratio -from_partition {* 0.3} -to_partition {* 0.3}
```

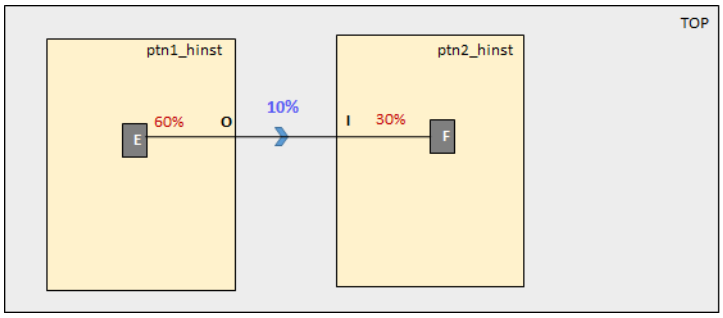
Note: * matches with any partition hinst. However, it does not matches with TOP.
Following is the output:



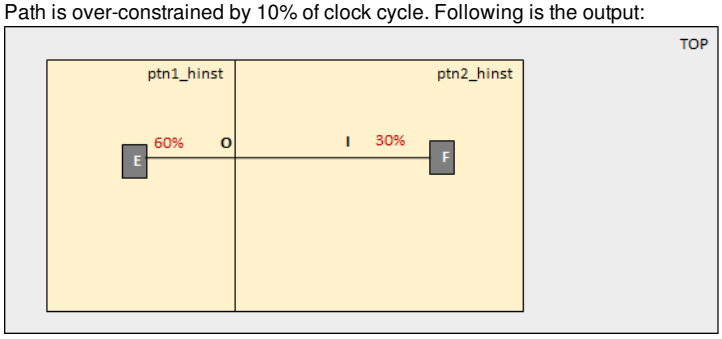
- The following command presents an example of the PTN to PTN path:

```
set_cycle_budget_ratio -from_partition {ptn1_hinst 0.6} -to_partition {ptn2_hinst 0.3}
```

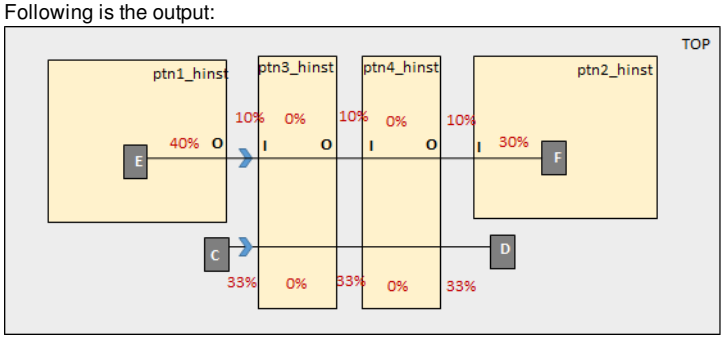
Following is the output:



- The following command presents an example of the PTN to PTN Abutted path:
  ```
  set_cycle_budget_ratio -from_partition {ptn1_hinst 0.6} -to_partition {ptn2_hinst 0.3}
  ```
  Path is over-constrained by 10% of clock cycle. Following is the output:



- The following command presents an example of how the top level budget is split into top and FT segments of the paths:
  ```
  set_cycle_budget_ratio -from_partition {ptn1_hinst 0.4} -to_partition {ptn2_hinst 0.3}
  ```
  Note: FT segment is over-constrained and given 0% budget.
  Following is the output:



Related Information

The following sections in the Timing Budgeting chapter of the *Innovus User Guide*:

- "Deriving Timing Budgets"
- "Budgeting Using Text Commands"
- "Top-Level Budgets Derived by Using Active Logic View"
- "Deriving Preliminary Budgets in Early Design Phase"
- "Budgeting Output Files for MMMC Designs"
- "Analyzing Timing Budgets"
- "Resolving Conflicts with Path-Based Exceptions"
- "Customizing Budget Generation"
- "Verifying Timing Budgets"

- "Reading the Justify Budget Report"
- "Warning Report"

set_fixed_budget

```
 set_fixed_budget
[-help] {[[[{-from {list_of_pins} | -from_flops_of {hinst}} {-to {list_of_pins} | -to_flops_of {hinst}}] | [-through {list_of_hinst}]] {-
slack | -delay } [-add ] value] | [-reset ]}
```

This command fixes the budget for a particular path segment between partitions and start/end points and helps in reducing the distribution budget delay in the path segments.

Note: Do not use this command with cycle-based timing budgeting.

This command is valid only for the following segments:

- Chip In -> Partition In
- Start Point -> Partition In
- Partition In -> Partition Out
- Partition In -> End Point (FF D Pin) inside same partition
- Start Point (FF Q Pin) inside same partition -> Partition Out
- Partition Out -> Chip Out
- Partition Out -> End Point

Parameters

`-help`	Prints out the command usage.
`-add`	Adds the slack/delay to previous value. *Data_type*: bool, optional
value	Presents delay or slack values. *Data_type*: float, required
`-delay`	Specifies the delay value. *Data_type*: bool, optional
`-from {list_of_pins}`	Presents the 'from pins' to be analyzed. *Data_type*: string, required
`-from_flops_of {hinst}`	Presents the 'from hierarchical inst' for which start pins need to be analyzed. *Data_type*: string, required
`-reset`	Resets all previously declared fixed budgets. *Data_type*: bool, optional
`-slack`	Specifies the slack value. *Data_type*: bool, optional
`-through {list_of_hinst}`	Specifies the hierarchical instances to be analyzed. *Data_type*: string, required
`-to {list_of_pins}`	Specifies the 'to pins' to be analyzed. *Data_type*: string, required
`-to_flops_of {hinst}`	Specifies the 'to hierarchical inst' for which end pins need to be analyzed. *Data_type*: string, required

Examples:

The following command fixes the budget of the hierarchical instance feed-through paths to the existing delay:

```
set_fixed_budget -through P5 -slack 0
```

The following command gives a fixed clack of 1 to the paths from port 'in' to hierarchical instance input pin 'p4/in'; budgets will distribute rest of the slack to other segments of the paths:

```
set_fixed_budget -from in -to P4/in -slack 1
```

The following command fixes the delay of this segment to 0.5 from the hierarchical instance input pin 'p4/in' to the endpoint 'P4/FF1/D'; budgets will be calculated for the whole path considering the path delay of this segment as 0.5:

```
set_fixed_budget -from P4/in -to P4/FF1/D -delay 0.5
```

References:

See the 'Fixing Budget' section in the Timing Budgeting chapter of the *Innovus User Guide* document for more details.

set_partition_user_constraints_file

```
set_partition_user_constraints_file
[-help]
[-name name]
[-partition partitionName]
[-view]
```

Specifies the constraints file for a specific partition. The software appends the constraint information in this file to the timing constraints generated when you use the `write_partitions` or `write_timing_budgets` command. You can specify the following constraints in the constraints file for the partition:

- `set_clock_uncertainty`
- `set_clock_latency` (network latency constraints)

You must specify the `-budget_local_latency` and `-budget_local_uncertainty` parameters in the `set_db` command before using the `set_partition_user_constraints_file` command.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `set_partition_user_constraints_file` parameter. For a detailed description of the command and all its parameters, use the man command: `man set_partition_user_constraints_file`
`-name` *name*	Specifies the name of the file containing constraints specific to a partition. *Data_type*: string, required
`-partition` *partitionName*	Specifies the name of the existing partition. *Data_type*: string, required
`-view`	Specifies a partition-specific constraint file for each MMMC view. *Data_type*: string, required

Related Information

The Timing Budgeting chapter of the *Innovus User Guide* document.

update_timing_budget

```
update_timing_budget
[-help]
{-partition partitionName | -hinst instanceName}
-in_file fileName
[-setup | -hold]
[-view viewName]
```

Modifies budgets after they have been saved using the `write_timing_budgets` command. To modify budgets, you must create an input budget file containing constraints that look like an SDC file. It only supports `set_input_delay` and `set_output_delay` constraints. The clock names in these constraints are same as the the ones in the generated budget file. You can provide a bus name instead of a pin name to apply the same budget to all pins of the bus.

Parameters

`-help`	Prints a brief description that includes the type and default information for each `update_timing_budget` parameter. For a detailed description of the command and all its parameters, use the man command: `man update_timing_budget`
`-in_file fileName`	Specifies the name of the input budget file containing manual budgets. *Data_type*: string, required
`-hold`	Specifies to modify the hold information. *Data_type*: bool, optional
`-hinst instanceName`	Specifies the name of the hierarchical instance for which manual budgets are to be applied. *Data_type*: string, required
`-partition partitionName`	Specifies the name of the partition for which manual budgets are to be applied. *Data_type*: string, required
`-setup`	Specifies to modify the setup information. *Data_type*: bool, optional
`-view viewName`	Modifies the budget for the specified view. If you derived the timing budget in MMMC mode (default), this parameter is required to modify the budget. *Data_type*: string, optional

Example

The following command modifies the budget data for the instance inst1:

```
update_timing_budget -hinst inst1 -in_file mod_inst
```

Related Information

The 'Modify Budgets' section in the Timing Budgeting chapter of the *Innovus User Guide* document.

write_timing_budgets

```
write_timing_budgets
[-help]
[-dir dirName]
[-hinsts string | -partitions string]
[{instList}]
[-module_model_tag string]
```

Saves the previously derived timing budgets to a specified directory. The main outputs of this command are the block constraints and the timing model. In the MMMC mode, the `write_timing_budgets` outputs the block and top `viewdefinition.tcl` as well.

Use this command after deriving the time budgets using the `create_timing_budget` command.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `write_timing_budgets` parameter. For a detailed description of the command and all its parameters, use the man command: `man write_timing_budgets`
instList	Saves the budget files for instances. You use this parameter with the `create_timing_budget -hinsts` command. *Data_type*: string, optional
`-dir` *dirName*	
	Specifies the relative directory name where the derived timing budget files will be saved. These files are saved in a sub-directory named `cellName_timingViewName.sdc.gz` at the `dirName/cellName/cellName_timingViewName.sdc.gz` location through the relative directory path. If you do not specify this parameter, the timing budget files are saved in the current directory. For example, if you run the following command, a directory named `STB` will be saved at the current path: `write_timing_budgets -dir STB` *Data_type*: string, optional
`-hinsts` *instList* `|` `-partitions` *ptnList*	
	Saves the budget file for partitions or instances for which you have generated timing budgets. You do not need to specify `-partitions` or `-hinsts` if you have previously specified `create_timing_budget -ilm`. The `-partitions` parameter saves the budget files for partitions. You use this parameter with the `create_timing_budget -partitions` command. The `-hinsts` parameter saves the budget files for instances. You use this parameter with the `create_timing_budget -hinsts` command. *Default:* The tool saves budgets for all instances and partitions specified by `create_timing_budget`. *Data_type*: string, optional
`-module_model_tag` *string*	
	Creates a new `mmmc` directory in the tag specified for Integrated Hierarchical Database (iHDB) file structure and saves the view definitions, libs, and constraints therein. The created `mmmc` directory will be located as follows: *ihdb_root*/*block_name*/*tag_name*/mmmc For example: • The `write_timing_budgets -module_model_tag ABCD` command will save view definition, libs, and constraints within the `ihdb_root/block_name/ABCD/mmmc` directory. • The default behavior of the `write_partitions -module_model_tag ABCD` command is not impacted, and the view definition, libs, and constraints are saved in the `.enc.dat` file within the `ihdb_root/block_name/ABCD/pnr` directory. • The `write_timing_budgets -module_model_tag ABCD` and `write_partitions -module_model_tag ABCD` commands are completely independent (those will not affect the functioning of each other). *Data_type*: string, optional

Examples

The following examples are based on a scenario, where a design with the `ptn1` partition has two instances, `L1` as the master, and `L1_clone` as the clone:

- The following set of commands saves the timing budget files for partition in the `STB/ptn1` directory. It saves the timing budget information for both the partition and top. Also, since it uses the `master_only` mode, the budgets will be based on the master timing only:

```
create_timing_budget -partitions ptn1
write_timing_budgets -dir STB
```

- The following set of commands saves timing budget files only for the `L1_clone` instance in the `STB/ptn1` directory. Since it uses the `merged` mode, the timing budgeting information will be based on the worst case data from both the instances. In this case, any of the master or clone instances, or both, can be specified while running `write_timing_budgets`. The best approach is to not specify either of those, and let the `write_timing_budgets` command save the information generated by the `create_timing_budget` command:

```
set_db budget_master_clone merged
create_timing_budget -hinsts {L1 L1_clone}
write_timing_budgets -dir STB -hinsts {L1_clone}
```

- The following set of commands saves the timing budget files for the `L1` instance in the `STB/ptn1` directory. It saves the timing budget information for only the partition instance:

```
create_timing_budget -hinsts L1 (or L1_clone)
write_timing_budgets -dir STB
```

- The following set of commands saves the timing budget files for the `L1` instance in the `STB/ptn1.L1` directory, and for the `L1_clone` instance in the `STB/ptn1.L1_clone` directory. It saves the timing budget information only for the partition instances:

```
create_timing_budget -hinsts {L1 L1_clone}
write_timing_budgets -dir STB
```

- The following set of commands saves the timing budget files only for the `L1_clone` instance in the `STB/ptn1.L1_clone` directory:

```
create_timing_budget -hinsts {L1 L1_clone}
write_timing_budgets -dir STB -hinsts {L1_clone}
```

Related Information

The following sections in the Timing Budgeting chapter of the *Innovus User Guide* document:

- Deriving Timing Budgets
  - Budgeting Using Text Commands
  - Top-Level Budgets Derived by Using Active Logic View
  - Deriving Preliminary Budgets in Early Design Phase
- Budgeting Output Files for MMMC Designs
- Analyzing Timing Budgets
- Customizing Budget Generation
- Verifying Timing Budgets
- Reading the Justify Budget Report
- Warning Report

39

Timing Constraint Commands

- create_clock
- create_generated_clock
- current_design
- current_instance
- get_current_constraint_mode
- group_path
- reset_annotated_check
- reset_annotated_delay
- reset_annotated_transition
- reset_aocv_stage_weight
- reset_case_analysis
- reset_clock
- reset_clock_gating_check
- reset_clock_groups
- reset_clock_latency
- reset_clock_sense
- reset_clock_transition
- reset_clock_tree_latency
- reset_clock_uncertainty
- reset_data_check
- reset_disable_clock_gating_check
- reset_disable_timing
- reset_drive
- reset_driving_cell
- reset_generated_clock
- reset_ideal_latency
- reset_ideal_network

- reset_ideal_transition
- reset_input_delay
- reset_load
- reset_max_capacitance
- reset_max_fanout
- reset_max_time_borrow
- reset_max_transition
- reset_min_capacitance
- reset_min_fanout
- reset_min_pulse_width
- reset_min_transition
- reset_mode
- reset_output_delay
- reset_path_exception
- reset_path_group
- reset_pll_timing
- reset_propagated_clock
- reset_property
- reset_pulse_clock_max_transition
- reset_pulse_clock_max_width
- reset_pulse_clock_min_transition
- reset_pulse_clock_min_width
- reset_resistance
- set_active_clocks
- set_annotated_check
- set_annotated_delay
- set_annotated_transition
- set_aocv_interface_path_offset
- set_aocv_stage_weight
- set_case_analysis
- set_clock_exclusivity
- set_clock_gating_check

- set_clock_groups
- set_clock_latency
- set_clock_sense
- set_clock_transition
- set_clock_uncertainty
- set_data_check
- set_disable_clock_gating_check
- set_disable_timing
- set_dont_touch
- set_dont_touch_network
- set_dont_use
- set_drive
- set_driving_cell
- set_false_path
- set_fanout_load
- set_ideal_latency
- set_ideal_network
- set_ideal_transition
- set_input_delay
- set_input_transition
- set_load
- set_logic_one
- set_logic_zero
- set_max_capacitance
- set_max_delay
- set_max_fanout
- set_max_time_borrow
- set_max_transition
- set_min_capacitance
- set_min_delay
- set_min_fanout
- set_min_pulse_width

- set_min_transition
- set_mode
- set_multicycle_path
- set_normalized_driver_waveform_lib
- set_output_delay
- set_pll_timing
- set_propagated_clock
- set_property
- set_pulse_clock_max_transition
- set_pulse_clock_max_width
- set_pulse_clock_min_transition
- set_pulse_clock_min_width
- set_resistance
- write_annotated_transition
- set_sense
- reset_sense

create_clock

```
create_clock
-period period_value
[-name clock_name]
[-waveform edge_list]
[-add]
[-comment string]
[source_objects]
```

Creates a clock object and defines its waveform in the current design. If you do not specify any source objects, but you specify the clock name then a virtual clock is created. The new clock has an ideal clock latency, and the software does not assume any propagation delay through the clock network.

The clock that you create using the `create_clock` command cannot have multiple pulses per period. You can specify multiple constraints on the same pin using the `-add` parameter.

For example, if you specify the following command sequence, the `PH2` clock waveform overwrites the `PH1` waveform.:

```
create_clock -name PH1 ... [get_ports sysclk]
create_clock -name PH2 ... [get_ports sysclk]
```

To have both clocks applied, you must specify the following command sequence:

```
create_clock -name PH1 ... [get_ports sysclk]
create_clock -name PH2 -add ... [get_ports sysclk]
```

You can create a collection of clocks, which is a group of objects referenced by a string identifier using the `get_clocks` command.

Parameters

`-add`	Specifies multiple clocks on the same source for simultaneous analysis with different clock waveforms. You must specify the `-name` parameter with this parameter. If one of the *`source_objects`* is already the source of a clock, the source is removed from that clock. If you need to specify multiple clocks on the same source for simultaneous analysis, you must specify the `-add` parameter, to add this clock to the existing clock.
`-comment` *`string`*	Allows you to insert comments. The comments are read but are ignored with a one-time warning message. They are not used during timing analysis reporting nor are they written out during saving of constraints.
`-name` *`clock_name`*	Specifies the string name of the ideal clock being created. If you do not specify this option, the clock gets the same name as the first clock source that you specify using the `-source_objects` option. If you do not specify any source_objects, you must specify the `-name` *`clock_name`* option, which creates a virtual clock not associated with a port or pin.
`-period` *`period_value`*	Specifies the value of the ideal clock in library time units.
`source_objects`	Specifies the list of pins or ports associated with an ideal clock waveform of the clock. If you do not specify this option, you must specify the `-name` *`clock_name`* option, which creates a virtual clock not associated with a port or pin. If you specify a clock on a pin that already has a clock, the new clock replaces the old one.
`-waveform` *`edge_list`*	Specifies the rise and fall edge times of the clock waveforms in library time units over an entire clock period. The first value in the edge list is the time at which the first transition on the clock occurs, which is the first rising transition after time zero, and edges are monotonically increasing. The numbers should represent one full clock period. If you do not specify the `-waveform` *`edge_list`* option, a default waveform is assumed, and the leading edge is placed at `0` and the trailing edge is placed at the midpoint of the period, such that a symmetric clock is generated.

Examples

- The following command creates an `ideal_clk1` clock on port `clkA` with a period of `5`, a rise at `0`, and a fall at `2.5`:
  ```
  create_clock [get_ports {clkA}] -name ideal_clk1 -period 5 -waveform {0 2.5}
  ```
- The following command creates an `ideal_clk2` clock on port `clkB` with a period of `10`, a rise at `1`, and a fall at `9`:
  ```
  create_clock [get_ports {clkB}] -name ideal_clk2 -period 10 -waveform {1 9}
  ```

Related Information

- all_clocks
- get_clocks
- get_pins
- get_ports
- set_clock_latency
- report_clocks
- set_clock_uncertainty
- set_input_delay
- set_output_delay

create_generated_clock

```
create_generated_clock
[-help]
target_pin_list
[-add]
[-comment string]
[-duty_cycle percent]
[-edge_shift edge_shift_list]
[-invert]
[-master_clock source_clock_name]
[-name <clock_name>]
-source source_pin
{[-multiply_by factor]
[-divide_by factor]
[-edges edge_list]
[-combinational]}
```

Creates a new clock signal from the clock waveform of a given pin in the design, and binds it with the pins or hierarchical pins in the *target_pin_list* argument. Whenever the source clock changes, the derived clock(s) change automatically.

You can generate a new clock waveform using one of the three ways:

- Multiply the frequency of the source clock
- Divide the frequency of the source clock
- Select the edges of the source clock to be mapped to the edges of the new clock

Given below are the SDC semantics to be used when the `create_generated_clock` command is defined:

- The edges are interpreted as alternating rise and falling edges.
- Each edge must be greater than and equal to its previous edge.
- Less than 3 edges are not allowed.
- Legal edge numbers start at “1”, edge 0 is undefined.
- Legal assertion of period 0 will be -edges {1 1 1 }.

If incorrect clock definitions are specified, an error will occur and the clock will not be generated.

When the `create_generated_clock` command is specified without the `-master_clock` parameter, and the software finds multiple clocks reaching the specified source pin, then the regular clocks are picked ahead of the generated clocks. In case if multiple regular clocks are present, the master clock is picked based on alphabetical sorting order of the clock names.

Note: A `create_generated_clock` assertion on a pin overrides any existing `create_generated_clock` assertion if both the constraints have the same source pin. In the following command sequence, the second assertion overrides the first:

```
create_generated_clock -name GCK1 -source CK -divide_by 2 U_FD1/Q
create_generated_clock -name GCK2 -source CK -multiply_by 2 U_FD1/Q
```

To have both assertions applied, you must specify the `-add` parameter. For example:

```
create_generated_clock -name GCK1 -source CK -divide_by 2 U_FD1/Q
create_generated_clock -name GCK2 -add -master_clock CLK -source CK -multiply_by 2    U_FD1/Q
```

Parameters

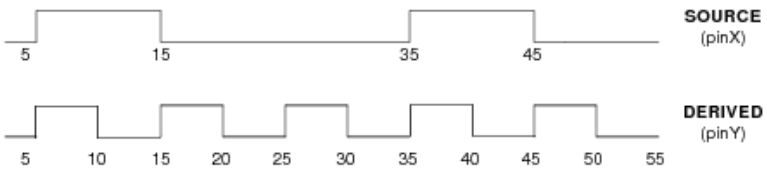
`-help`	Prints out the command usage.
`-add`	Models multiple generated clocks on the same source when multiple clocks must fan into the source pin. Ideally, one generated clock must be specified for each clock that fans into the master pin. Specify this parameter with the `-name` and `-master_clock` parameters. You can use the `-add` parameter to specify a different clock name for each generated clock, when used with the `-master_clock` option. Subsequently, you can use this clock name for setting other constraints, such as, `set_false_path` or `set_input_delay`.
`-combinational`	Restricts the path search for latency to only paths which do not cross sequential devices, or the source pins of other generated clocks. When you specify `-combinational`, if there are both sequential and combinational latency paths, the software only considers the combinational paths for the generated clock. If there are no combinational latency paths for the generated clock, the generated clock does not inherit latency from its parent clock hierarchy. By default, when calculating the latency of a generated clock, the software searches for the longest (and shortest) paths from the parent waveform to the generated clock point. All paths are considered, including those which traverse sequential elements such as flip-flops, registers, and latches. The default behavior is intended for cases where there is a cascade of generated clocks, and some of the clocks are not explicitly defined. The `-combinational` option can be used in conjunction with the `-multiply_by`, `-divide_by`, or `-edges` options.
`-comment` *string*	Allows you to insert comments. The comments are read but are ignored with a one-time warning message. They are not used during timing analysis reporting nor are they written out during saving of constraints.
`-divide_by` *integer*	Determines the frequency of the new clock by dividing the frequency of the source clock by *dfactor*. For example, if *dfactor=5*, the new frequency is 1/5 of the source frequency, and the new clock period is 5 times the source clock period.
`-duty_cycle` *percent*	Sets the duty cycle (high pulse width or clock period). The number (*percent*) is between `0` and `100`. For example, if *percent=50*, the high pulse width of the new clock waveform is the same as its low pulse width. **Note:** The `-duty_cycle` option can only be used with the `-multiply_by` option, otherwise it will be ignored with a warning.
`-edges` *edge_list*	Selects a list of edges from the source clock that form the edges of the derived clock. See the frequency division example below.

`-edge_shift` *`edge_shift_list`*	Specifies the amount of shift for each edge in the *`edge_list`* option.
`-invert`	Inverts the source clock. Use with the `-source` option and the `-multiply_by` or `-divide_by` options.
`-master_clock` *`source_clock_name`*	Generates a target clock from the specified clock name. If you specify the `-master_clock` option and multiple signals arrive at the source pin, the clock specified by this option is used to generate the target clock. Use the `-add` parameter along with`-master_clock` parameter to add a clock to the existing clock when you want to specify multiple clocks on the same source for simultaneous analysis.
`-multiply_by` *`integer`*	Determines the frequency of the new clock by multiplying the frequency of the source clock by *`mfactor`*. For example, if *`mfactor=4`*, the new frequency is 4 times the source frequency, and the new clock period is 1/4 of the source clock period.
`-name` *`clock_name`*	Specifies the name for the generated clock. *Default*: The system creates a name for you. Always specify a name for the generated clock. Some commands issue an error if you use the system-generated name.
`-source` *`source_pin`*	Specifies the name of the source pin from which the new clock is to be derived. This is a mandatory option.
`target_pin_list`	Specifies the list of pins where the generated clock assertion is applied. The clock signal propagating downstream from each of these pins becomes associated with the generated clock.

Examples

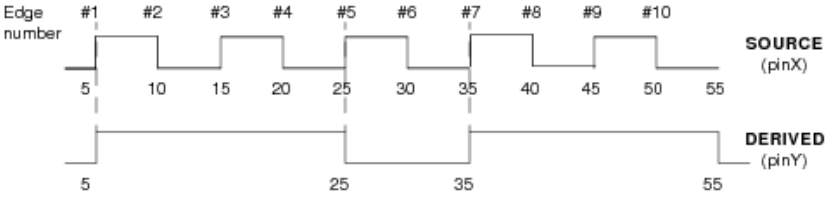
- The following command generates the derived clock waveform, as shown in Figure 36-1: create_generated_clock -name mult_3 -source pinX -multiply_by 3 -duty_cycle 50 pinY

Figure 36-1 **Derived Clock (Frequency Multiplication)**



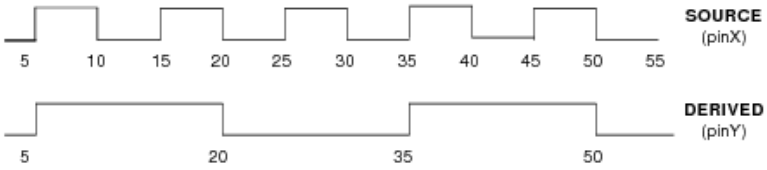
- The following command shows one way to create a divide-by-3 clock by using edge numbers of the source waveform to specify the new waveform, as shown in Figure 36-2,: create_generated_clock -name genclk -source pinX -edges {1 5 7} pinY

Figure 36-2 **Derived Clock (Frequency Division)**



- The following command uses the `-divide_by` option to get the same clock period; the difference is in the duty cycle: create_generated_clock -name div_3 -source pinX -divide_by 3 pinY
  With the `-edges` option, the above example creates a clock with a duty cycle of 67 percent (20/30). However, using the `-divide_by 3` option preserves the 50 percent duty cycle of the source clock as shown in the figure below.

Figure 36-3 **Derived Clock (Frequency Division, 50 Percent Duty-cycle)**



- Using the following commands, when there is more than 1 source clock, the tool creates the generated clock based on the source clock specified with the `-master` option: create_clock -name c1 -period 10 [get_ports {ck}] -add create_clock -name c2 -period 10 [get_ports {ck}] -add create_generated_clock -name newclk -source ck -divide_by 2 -master_clock c1 reg/Q -add
  Using the `report_clocks` command, the output waveform of the `newclk` clock is a divide-by-2 of the `c1` clock and is reported as shown in example below.

report_clocks after using the -master option

```
+-------------------------------------+
|         Clock Descriptions          |
|-------------------------------------|
|  Clock |  Period |  Lead  |  Trail  |
|  Name  |         |        |         |
|--------+---------+--------+---------|
|   c1   | 10.0000 | 0.0000 |  5.0000 |
|   c2   | 10.0000 | 0.0000 | 5.0000  |
| newclk | 20.0000 | 0.0000 | 10.0000 |
+-------------------------------------+
```

Related Information

- `create_clock`
- `get_clocks`
- `report_clocks`

current_design

```
current_design
[-help]
[<design_name>]
```

Sets the specified design to be the current design.

Parameters

`-help`	Prints out the command usage.
design_name	Specifies the name of the top design.

Related Commands

- set_max_transition
- set_min_capacitance
- set_min_transition

current_instance

```
current_instance
[hinst]
```

Sets the instance given by *hinst* to be the current instance. Design objects referenced by subsequent commands can be found relative to this instance. All searches for design objects are started in this instance. If design objects are referenced hierarchically, *hinst* is used as the root (top) of the hierarchy.

Parameters

hinst	Specifies the name of the instance to become the current instance. If *hinst* is not given, the current instance is set to the top instance of the hierarchy.

Examples

- The following command sets the current instance to MAC/MULTI, sets a constant on a register pin inside the hierarchical instance and then later sets the current instance back to top:

```
current_instance MAC/MULT1
set_case_analysis 0 [get_pins {reg6/SE}]
...
current_instance
```

Related Commands

- set_case_analysis

get_current_constraint_mode

```
get_current_constraint_mode
[-help]
```

Determines the constraint mode that is being currently read.

This command can be used inside SDC constraint files which are shared across multiple constraint modes to conditionalize how the constraints are interpreted for specific modes.

group_path

```
group_path
[-help]
[-comment string]
-name path_group_name
[[-from from_list]
 [-to to_list]
 [-through through_list]]
```

Groups paths in a design, and identifies them with a path group name. You can use the path group names for reporting.

You cannot use group_path statements in the SDC file when the software is in MMMC mode. In this case a warning is issued. However, the software supports the group_path command in MMMC mode as a command outside of the SDC file. The path groups are considered to be global in nature and apply to all the views, and are not supported on MMMC mode-specific basis. The path groups are persistent through a saveDesign to restoreDesign sequence. They are saved in the `<saveDir>/mmmc/pathgroup.sdc` file of a MMMC save directory.

Note that if a path can be included in multiple path groups, it will only be included in the path group with the most specific definition (`-from ... -through ... -to ...` is more specific than `-from ... -to ...` is more specific than `-from ...`).

Parameters

`-help`	Prints out the command usage.
`-comment string`	Allows you to insert comments. The comments are read but are ignored with a one-time warning message. They are not used during timing analysis reporting nor are they written out during saving of constraints.
`-from from_list`	Specifies a list of pins, instances, or clocks that are the start of the paths. When you specify a clock object, all paths that are triggered by the clock are included in the path group.
`-name path_group_name`	Specifies the name of the path group. If the path_group_name already exists, the specified paths are added to the existing path group.
`-through through_list`	Specifies the pins or instances where the paths go through. When you specify multiple through list, the paths that go through the objects in the specified order are added to the path group.
`-to to_list`	Specifies a list of pins, instances, or clocks where the path ends. You must specify valid endpoints in the *to_list* or else the software ignores the constraint. When you specify a clock object, all paths that are checked by the clock are included in the path group. If you specify an instance name in the *to_list*, all the data input pins of the instance are added to the path group.

Example

- The following example groups paths that start from specific pins and end at the D pin of any instances whose name start with the string ff. The example also reports the specified path groups:

```
group_path -name path1 -from {ff1/CLK ff2/CLK} -to {ff*/D}
report_timing -path_group path1
```

- The following command groups paths that go through specific points.

```
group_path -name path2 -through an1/Z -through an2/Z -through an3/Z
```

Related Commands

- report_timing
- get_path_groups
- reset_path_group
- report_path_groups
- create_basic_path_groups

reset_annotated_check

```
reset_annotated_check
[-clock [rise | fall]]
[-cond sdf_condtion]
[-setup | -recovery]
[-hold | -removal]
[-nochange_high | -nochange_low]
[-rise] [-fall]
{[[-from from_pins][-to to_pins]] | -all}
[obj_list]
```

Removes annotated timing check values that you had set by using the `set_annotated_check` command or by SDF annotation.

Note: The `reset_annotated_check` command does not reset the timing check values set by SDF annotations.

Parameters

`-all`	Resets all annotated timing check values in the design.
`-clock [rise | fall]`	Specifies the direction of the related pin of the check to reset. *Default:* Resets checks against both edges of the clock
`-cond sdf_condtion`	Resets all annotated timing check values for which a condition is specified.
`-fall`	Restricts the reset to only affect the falling edge of the constrained pins. *Default:* Checks on both rising and falling signals are reset
`-from from_pins`	Specifies the beginning instance pin of the timing arcs for which the annotated check is reset. This is usually the reference (for example, clock pin) of the constraint. The selected timing arcs are those that have their source pins specified using this option and the sink pins specified using the `-to` option. If this option is not specified, only timing arcs whose sink pins are specified using the `-to` option are selected. You cannot use this parameter with *`obj_list`* parameter.
`-hold | -removal`	Resets annotations on the hold or removal check type.
`-nochange_high | -nochange_low`	Resets the annotations on the specified timing check type.
`obj_list`	Resets the annotated checks on specified list or collection of cell instances.

`-rise`	Restricts the reset to only affect the rising edge of the constrained pins. *Default:* Checks on both rising and falling signals are reset
`-setup | -recovery`	Resets annotations on the setup or recovery check type.
`-to` *`to_pins`*	Specifies the terminating instance pin of the timing arcs for which the annotated check is reset. This is usually the constrained pin (data pin) of the check. The selected timing arcs are those that have their sink pins specified using this option and the source pins specified using the `-from` option. If this option is not specified, only timing arcs whose source pins are specified using the `-from` option are selected. You cannot use this parameter with *`obj_list`* parameter.

Examples

- The following command resets all the timing check annotations in the design:
  reset_annotated_check -all
- The following command resets all timing check annotations on a particular cell:
  reset_annotated_check [get_cells BLK/BR1]
- The following command resets specific checks on a specific instance:
  reset_annotated_check -setup -hold [get_cells BLK/BR2]
- The following command resets checks on a specific timing check arc:
  reset_annotated_check -from BLK/BR1/CK -to BLK/BR1/D

Related Commands

- `set_annotated_check`

reset_annotated_delay

```
reset_annotated_delay
{[-all] | [[-from from_pins] [-to to_pins]] | [object_list]}
```

Resets annotated delay assertions that have been previously specified using the `set_annotated_delay` command. When you reset an annotated delay, the system uses the delays which are calculated from the original libs for the arc.

Parameters

`-all`	Resets all annotated delay assertions in the design.
`-from` *from_pins*	Specifies timing arcs for which the delay is reset. The selected timing arcs are those that have their source pins specified using this option and the sink pins specified using the `-to` option. If this option is not specified, only timing arcs whose sink pins are specified using the `-to` option are selected.
object_list	Resets the annotated delays on specified list or collection of cell instances.
`-to` *to_pins*	Specifies timing arcs for which the delay is reset. The selected timing arcs are those that have their sink pins specified using this option and the source pins specified using the `-from` option. If this option is not specified, only timing arcs whose source pins are specified using the `-from` option are selected.

Examples

- The following command resets all the annotated delay assertions in the design:
  reset_annotated_delay -all
- The following command resets all the `set_annotated_delay` assertions applied on pins (either `from` or `to`) of instance `f0`:
  reset_annotated_delay f0
- The following command resets the `set_annotated_delay` assertions with the `from` and `to` pin pair combination:
  reset_annotated_delay -from u1/Y -to u2/A

Related Information

- `set_annotated_delay`

reset_annotated_transition

```
reset_annotated_transition
{-all | port_or_pin_list}
```

Resets annotated transitions previously asserted by the `set_annotated_transition` command.

Parameters

`-all`	Resets all annotated transitions in the design.
`port_or_pin_list`	Specifies a list of specific pins or primary ports to reset the transition.

Examples

- The following commands reset the `set_annotated_transition` constraint on an internal pin `pad/Y0`.
  ```
  reset_annotated_transition pad/Y0
  ```
  or
  ```
  reset_annotated_transition [get_pins {pad/Y0}]
  ```
  You can also specify a collection as the pin argument. For example:
  ```
  reset_annotated_transition [get_pins pad/*]
  ```
- The following commands reset the `set_annotated_transition` constraint set on a port `in`:
  ```
  reset_annotated_transition in
  ```
  or
  ```
  reset_annotated_transition [get_ports {in}]
  ```
  Any `set_input_transition` constraint applied on the same port is not removed. This is applicable only for input ports, as `set_input_transition` cannot be applied on output ports.
- The following command resets all the `set_annotated_transition` constraints:
  ```
  reset_annotated_transition -all
  ```

Related Information

- `set_annotated_transition`

reset_aocv_stage_weight

```
reset_aocv_stage_weight
port_list
[-early]
[-late]
```

Resets the existing stage weight values set for the specified ports.

Parameters

`-early`	Resets weight for early paths.
`-late`	Resets weight for late paths.
port_list	Specifies a list or collection of input, output, and bidi ports for which stage weight values are to be reset.

Examples

- The following example resets the stage weight values of ports `clock1` and `out0`:
  reset_aocv_stage_weight [get_ports {clock1 out0}]

Related Information

- `set_aocv_stage_weight`

reset_case_analysis

```
reset_case_analysis
port_pin_list
```

Removes the assertions set by the set_case_analysis command.

Parameters

port_pin_list	Specifies the list of ports or pins for which the assertions are to be removed.

Examples

- The following commands sets the assertion to 0 and then resets them:

```
set_case_analysis 0 CM1/S0
get_property [get_pins CM1/S0] constant_value

# Returns a value of 0

reset_case_analysis [get_pins CM1/S0]
get_property [get_pins CM1/S0] constant_value

# Returns a value of NA
```

Related Commands

- set_case_analysis

reset_clock

```
reset_clock
{-all | clock_list}
```

Removes previously created clock assertions.

Parameters

`-all`	Removes all clocks in the design.
`clock_list`	Specifies the list of clocks for which the clock assertion is removed.

Example

- The following commands reset a list of clocks and all clocks, and display the results using the `report_clocks` command:

```
report_clocks

# Generates the following report:

+-----------------------------------------+
| Clock Descriptions |
|-------------------------------------------------------------------|
| |                                    |  Attributes |
|-------+--------+--------+-------+-------+----------------------|
| Clock | Source | Period | Lead | Trail | Generated | Propagated|
| Name |         |        |      |       |           |           |
|-------+--------+--------+-------+-------+-----------+----------|
| CCLK |         | 10.000 | 0.000 | 5.000 | n         | n         |
| VCLK |         | 10.000 | 0.000 | 5.000 | n         | n         |
| WAVE |    tclk | 10.000 | 0.000 | 5.000 | n         | n         |
+-------------------------------------------------------------------+

reset_clock [get_clocks CCLK]
report_clocks

# Generates the following report:

+-----------------------------------------+
```

```
| Clock Descriptions |
|--------------------------------------------------------------|
|                                | Attributes |                |
|-------+--------+--------+-------+-------+----------------------|
| Clock | Source | Period | Lead | Trail | Generated | Propagated|
| Name  |        |        |      |       |           |           |
|-------+--------+--------+-------+-------+-----------+----------|
| VCLK  |        | 10.000 | 0.000 | 5.000 | n         | n         |
| WAVE  | tclk   | 10.000 | 0.000 | 5.000 | n         | n         |
+--------------------------------------------------------------+

reset_clock -all
report_clocks
```

This does not return anything indicating that all clocks were reset.

Related Information

- report_clocks

reset_clock_gating_check

```
reset_clock_gating_check
[-setup]
[-hold]
[-rise]
[-fall]
[-high | -low]
[obj_list]
```

Resets the specified setup and hold values for clock gating checks to the default values.

Parameters

`-high | -low`	Resets the non-controlling value (high or low) of the clock.
`-hold`	Resets clock gating hold values.
`-rise | -fall`	Resets the specified setup (or hold) value to the rising (or falling) signals on the specified pins.
`-setup`	Resets clock gating setup values.
obj_list	Specifies a list of clocks, instances, or pins for which clock gating checks are to be reset to default values. If the *obj_list* parameter is not specified, the clock gating check is reset on the current design.

Related Information

- set_clock_gating_check

reset_clock_groups

```
reset_clock_groups
[-help]
{-physically_exclusive | -logically_exclusive | -asynchronous}
{-name name_list | -all}
```

Resets clock groups that have been set using the `set_clock_groups` command.

Parameters

`-help`	Prints out the command usage.
`-all`	Removes all physically exclusive, logically exclusive, or asynchronous clock groups in the design. Removes all the physically exclusive, logically exclusive, or asynchronous clock groups in the design. This parameter should be used with one of the parameters: `-logically_exclusive`, `-physically_exclusive`, or `-asynchronous` at one time, and should not be used with the `-name` parameter.
`-asynchronous`	Specifies the asynchronous clock groups to be removed. This parameter must be used with either `-all` or `-name` parameter.
`-logically_exclusive`	Specifies the logically exclusive clock groups to be removed. This parameter must be used with either `-all` or `-name` parameter.
`-name` *name_list*	Specifies the previously defined names of clock groups to be removed. This parameter should be used with one of the parameters: `-logically_exclusive`, `-physically_exclusive`, or `-asynchronous` at one time.
`-physically_exclusive`	Specifies all the physically exclusive clock groups to be removed. This parameter must be used with either `-all` or `-name` parameter.

Examples

- The following command resets all physically exclusive clock groups in the design:
  ```
  reset_clock_groups -physically_exclusive -all
  ```
- The following command resets physically exclusive clock group `CLKGRP0`:
  ```
  reset_clock_groups -physically_exclusive -name CLKGRP0
  ```

Related Commands

- set_clock_groups

reset_clock_latency

```
reset_clock_latency
[-source]
[-clock clock_list]
obj_list
```

Resets assertions made by previous `set_clock_latency` commands.

Parameters

`-clock` *clock_list*	Specifies the list of clocks with which associated `set_clock_latency` constraints, at the pin level, are to be removed. **Note:** This parameter cannot be used with clocks specified using the *obj_list* parameter.
obj_list	Specifies a list of pins, ports, or clock objects for which the assertions are to be removed.
-source	Removes the source latency.

Examples

- The following commands show the use of the `reset_clock_latency` command to remove a leaf-level `set_clock_latency` assertion, after the clock tree has been put into propagated mode:

  ```
  set_clock_latency 3.141 [get_pins BLK/BR2/CK]
  report_timing -to BLK/BR2/D -path_type full_clock
  ```

  The following report is generated:

  ```
  Path 1: MET Setup Check with Pin BLK/BR2/CK
  Endpoint: BLK/BR2/D (v) checked with leading edge of 'WAVE'
  Beginpoint: BLK/BR1/Q (v) triggered by leading edge of 'WAVE'
  Other End Arrival Time 3.141 <---
  - Setup 0.191
  + Phase Shift 10.000
  = Required Time 12.950
  - Arrival Time 0.249
  = Slack Time 12.701
  ...

  set_propagated_clock [get_ports tclk]
  report_timing -path_type full_clock -net -to BLK/BR2/D
  ```

The following report is generated:

```
Path 1: MET Setup Check with Pin BLK/BR2/CK
Endpoint: BLK/BR2/D (v) checked with leading edge of 'WAVE'
Beginpoint: BLK/BR1/Q (v) triggered by leading edge of 'WAVE'
Other End Arrival Time 3.141 <--------
- Setup 0.191
+ Phase Shift 10.000
= Required Time 12.950
- Arrival Time 0.594
= Slack Time 12.356
```

...

In this example, setting the root of the clock tree to propagated mode is not sufficient to override the `set_clock_latency` assertion at the clock endpoint. You must use the `reset_clock_latency` command to remove the assertion, so that the register sees propagated clock timing.

```
reset_clock_latency [get_pins BLK/BR2/CK]
report_timing -path_type full_clock -net -to BLK/BR2/D
```

The following report is generated:

```
Path 1: MET Setup Check with Pin BLK/BR2/CK
Endpoint: BLK/BR2/D (v) checked with leading edge of 'WAVE'
Beginpoint: BLK/BR1/Q (v) triggered by leading edge of 'WAVE'
Other End Arrival Time 0.345 <--------
- Setup 0.186
+ Phase Shift 10.000
= Required Time 10.159
- Arrival Time 0.588
= Slack Time 9.571
```

- The following commands show that `reset_clock_latency` is only effective when it is applied on the same object for which you had set the assertion using the `set_clock_latency` command:

  Latency here is specified on a pin object:

  ```
  set_clock_latency 3.141 [get_pins BLK/BR2/CK]
  report_timing -path_type full -net -to BLK/BR2/D
  ```

  The following report is generated:

  ```
  Path 1: MET Setup Check with Pin BLK/BR2/CK
  Endpoint: BLK/BR2/D (v) checked with leading edge of 'WAVE'
  Beginpoint: BLK/BR1/Q (v) triggered by leading edge of 'WAVE'
  Other End Arrival Time 3.141 <-
  - Setup 0.191
  + Phase Shift 10.000
  = Required Time 12.950
  - Arrival Time 0.249
  ```

```
= Slack Time 12.701
```

Resetting of the latency is done on a clock object and has no effect on downstream latencies specified on pin objects.

In this example, a `reset_clock_latency` on the clock waveform that is driving BLK/BR2/CK will have no effect.

```
reset_clock_latency [get_clocks WAVE]
report_timing -path_type full -net -to BLK/BR2/D
```

The following report is generated:

```
Path 1: MET Setup Check with Pin BLK/BR2/CK
Endpoint: BLK/BR2/D (v) checked with leading edge of 'WAVE'
Beginpoint: BLK/BR1/Q (v) triggered by leading edge of 'WAVE'
Other End Arrival Time 3.141 <-
- Setup 0.191
+ Phase Shift 10.000
= Required Time 12.950
- Arrival Time 0.249
= Slack Time 12.701
```

Related Commands

- `set_clock_latency`

reset_clock_sense

```
reset_clock_sense
[-all]
| [[-clocks clock_list] pin_or_port_list]]
```

Resets clock sense constraint(s) specified using the `set_clock_sense` command.

Parameters

`-all`	Resets all the specified `clock_sense` constraints.
`-clocks` *`clock_list`*	Specifies the list of clocks for which the clock sense constraints are to be removed. The clocks must have previously been defined using the `set_clock_sense -clock` parameter.
`pin_or_port_list`	Specifies the list or collection of cell output pins for which the clock sense constraints will be removed.

Examples

- The following command resets all the previously defined `clock_sense` constraints from a design:

  ```
  reset_clock_sense -all
  ```

Related Commands

- `set_clock_sense`

reset_clock_transition

```
reset_clock_transition
clock_list
```

Removes the clock transition assertions you had set on the specified list of clock waveforms.

Parameters

clock_list	Specifies a list of clock waveforms for which the clock transition assertions are removed. The *clock_list* variable can be a Tcl list or collection of clock waveforms.

Examples

- The following commands set the clock transition assertions, and resets them and displays the results using the `report_timing` command:

```
set_clock_transition 0.123 [get_clocks WAVE]
report_timing -net -to BLK/BR2/D

# Generates the following report

Path 1: MET Setup Check with Pin BLK/BR2/CK
Endpoint: BLK/BR2/D (v) checked with leading edge of 'WAVE'
Beginpoint: BLK/BR1/Q (v) triggered by leading edge of 'WAVE'
Other End Arrival Time 0.000
- Setup 0.192
+ Phase Shift 10.000
= Required Time 9.808
- Arrival Time 0.249
= Slack Time 9.559
Clock Rise Edge 0.000
= Beginpoint Arrival Time 0.000
+-------------------------------------------------------------------------+
| Instance   | Cell    | Arc       | Pin     | Slew  | Delay | Arrival |
|            |         |           |         |       |       | Time    |
|------------+---------+-------------+-------+-------+-------+---------|
|            |         | tclk ^    | tclk    | 0.123 |       | 0.000 |
| CG/BC1     | BUFX2   |           | A       | 0.123 | 0.000 | 0.000 |
| CG/BC1     | BUFX2   | A ^ -> Y ^ | Y       | 0.123 | 0.000 | 0.000 |
```

```
| TC1         | BUFX2  |             | A       | 0.123 | 0.000 | 0.000 |
| TC1         | BUFX2  | A ^ -> Y ^ | Y       | 0.123 | 0.000 | 0.000 |
| CG1         | AND2X4 |             | B       | 0.123 | 0.000 | 0.000 |
| CG1         | AND2X4 | B ^ -> Y ^ | Y       | 0.123 | 0.000 | 0.000 |
| BLK/blkint | BUFX2  |             | A       | 0.123 | 0.000 | 0.000 |
| BLK/blkint | BUFX2  | A ^ -> Y ^ | Y       | 0.123 | 0.000 | 0.000 |
| BLK/BR1     | DFFHQX1|             | CK      | 0.123 | 0.000 | 0.000 |

reset_clock_transition [get_clocks WAVE]
report_timing -net -to BLK/BR2/D

# Generates the following report:

Path 1: MET Setup Check with Pin BLK/BR2/CK
Endpoint: BLK/BR2/D (v) checked with leading edge of 'WAVE'
Beginpoint: BLK/BR1/Q (v) triggered by leading edge of 'WAVE'
Other End Arrival Time 0.000
- Setup 0.191
+ Phase Shift 10.000
= Required Time 9.809
- Arrival Time 0.249
= Slack Time 9.560
Clock Rise Edge 0.000
= Beginpoint Arrival Time 0.000
+-------------------------------------------------------------------+
| Instance | Cell      | Arc         | Pin   | Slew  | Delay | Arrival |
|                                   |                       |   Time  |
|------------+---------+-------------+-------+-------+-------+---|
|            |         | tclk ^      | tclk  | 0.000 |       | 0.000 |
| CG/BC1     | BUFX2   |             | A     | 0.000 | 0.000 | 0.000 |
| CG/BC1     | BUFX2   | A ^ -> Y ^  | Y     | 0.000 | 0.000 | 0.000 |
| TC1        | BUFX2   |             | A     | 0.000 | 0.000 | 0.000 |
| TC1        | BUFX2   | A ^ -> Y ^  | Y     | 0.000 | 0.000 | 0.000 |
| CG1        | AND2X4  |             | B     | 0.000 | 0.000 | 0.000 |
| CG1        | AND2X4  | B ^ -> Y ^  | Y     | 0.000 | 0.000 | 0.000 |
| BLK/blkint | BUFX2   |             | A     | 0.000 | 0.000 | 0.000 |
| BLK/blkint | BUFX2   | A ^ -> Y ^  | Y     | 0.000 | 0.000 | 0.000 |
| BLK/BR1    | DFFHQX1 |             | CK    | 0.000 | 0.000 | 0.000 |
---------------------------------------------------------------------
```

Related Commands

- set_clock_transition

reset_clock_tree_latency

```
reset_clock_tree_latency
 [obj_list]
```

Resets all network clock latency set using the set_clock_latency command in the fanout of the specified clock or pins and ports. The reset_clock_tree_latency command resets the latency until the root pins of clocks and generated clocks. Use this command after building clock tree and setting the clock network from the root to propagated mode using the `set_propagated_clock` *`clockRootPinOrPort`* command.

Note: You can use the reset_clock_tree_latency command only on command line and command script. You cannot specify it in an SDC file.

Parameters

`obj_list`	Specifies list of objects - pins, ports, or clocks for which latency will be reset.

Related Commands

- `set_clock_latency`

reset_clock_uncertainty

```
reset_clock_uncertainty
[-setup]
[-hold]
[-half_cycle_jitter | -full_cycle_jitter]
{  {-from | -rise_from | -fall_from} clksig_from_list
   {-to | -rise_to | -fall_to} clksig_to_list
 | pin_or_clock_list
}
```

Removes the assertions that were made by previous `set_clock_uncertainty` commands.

Parameters

`{-from | -rise_from | -fall_from}` *clksig_from_list*	
	Removes the assertions for uncertainty value for the specified list of launching clocks. **Note:** You must specify one of these parameters with one of the `-to`, `-rise_to`, or `-fall_to` parameters. You cannot use these parameters when pins are specified.
	`-fall_from`: Removes the uncertainty value only from the falling edge of the launching clocks.
	`-from`: Removes the uncertainty value from both the rising and falling edges of the launching clocks.
	`-rise_from`: Removes the uncertainty value only from the rising edge of the launching clocks.
`-half_cycle_jitter | -full_cycle_jitter`	Removes or resets the clock source jitter assertion(s) of the specified type.
pin_or_clock_list	Removes the target uncertainty assertions from the specified list of clocks or pins. The uncertainty value is removed from the registers clocked by the target waveform, or register pins in the fanout of the specified pin. You can specify a list of clock names, or clock root pin or port names relative to the current module. You also can specify a collection of clocks, or clock root pins or ports. **Note:** You cannot use this parameter when any of the `-from` and `-to` parameter pairs are specified.
`-setup/-hold`	Specifies whether the clock uncertainty is removed for setup or hold checks. *Default*: Removes clock uncertainty for both setup and hold check.

`{-to | -rise_to | -fall_to}` *clksig_to_list*	
	Removes the uncertainty value from the specified list of capture clocks. **Note:** You must specify one of these parameters with one of the `-from`, `-rise_from`, or `-fall_from` parameters. You cannot use these parameters when pins are specified.
	`-fall_to`: Removes the uncertainty value only from the falling edge of the capture clocks.
	`-rise_to`: Removes the uncertainty value only from the rising edge of the capture clocks.
	`-to`: Removes the uncertainty value from both the rising and falling edges of the capture clocks.

Examples

The following commands sets and resets the clock uncertainty assertions and displays the results using the `report_timing` command:

- `set_clock_uncertainty 0.345 -from [get_clocks VCLK] -to [get_clocks WAVE]report_timing -net -from tin`

  ```
  # Generates the following report:
  Path 1: MET Setup Check with Pin TR1/BR1/CK
  Endpoint: TR1/BR1/D (v) checked with leading edge of 'WAVE'
  Beginpoint: tin (v) triggered by trailing edge of 'VCLK'
  Other End Arrival Time 0.000
  - Setup 0.192
  + Phase Shift 10.000
  - Uncertainty 0.345 <---
  = Required Time 9.463
  - Arrival Time 5.877
  = Slack Time 3.586
  ```

- `reset_clock_uncertainty -from [get_clocks VCLK] -to [get_clocks WAVE]report_timing -net -from tin`

  ```
  # Generates the following report:
  Path 1: MET Setup Check with Pin TR1/BR1/CK
  Endpoint: TR1/BR1/D (v) checked with leading edge of 'WAVE'
  Beginpoint: tin (v) triggered by trailing edge of 'VCLK'
  Other End Arrival Time 0.000
  - Setup 0.192
  + Phase Shift 10.000
  = Required Time 9.808
  - Arrival Time 5.877
  ```

```
= Slack Time 3.931
```

Related Information

- set_clock_uncertainty

reset_data_check

```
reset_data_check
{-from | -rise_from | -fall_from} pin_or_port_list
{-to | -rise_to | -fall_to} pin_or_port_list
[-setup]
[-hold]
[-clock clock_object]
```

Removes specified data-to-data checks that you have specified using the set_data_check command.

Parameters

`-clock` *clock_object*	Removes the data check with a related (`-from`) pin event that is triggered by the specified clock. Use this parameter only if the original `set_data_check` constraint also used the `-clock` parameter to specify an explicit clock relationship.
`-from` |`-rise_from` | `-fall_from` *pin_or_port_list*	Specifies the related pins of the data-to-data checks to be removed. The *pin_or_port_list* list can contain either object IDs or hierarchical names relative to the current module. Leaf design cells (instances) can also be specified. Use only one `-from`, `-rise_from`, or `-fall_from` parameter per command. By default, the `-from` parameter removes checks for both the rising and the falling edges. Using `-rise_from` parameter removes checks at the rising edge of the signal on the from pins. Using `-fall_from` parameter removes checks at the falling edge of the signal on the from pins.
`-setup/-hold`	Specifies whether the data check to be removed is for setup or hold check. If you do not specify either `-setup` or `-hold` option, both setup and hold checks are removed.
`-to` | `-rise_to` | `-fall_to` *pin_or_port_list*	Specifies the constrained pins of data-to-data checks to be removed. The *pin_or_port_list* list can contain either object IDs or hierarchical names relative to the current module. Leaf design cells (instances) can also be specified. Use only one `-to`, `-rise_to`, or `-fall_to` parameter per command. By default, the `-to` parameters removes both the rising and the falling edges. Using `-rise_to` parameter removes the rising edge of the signal on the to pins. Using `-fall_to` parameter removes the falling edge of the signal on the to pins.

Examples

- The following commands set and reset the data checks from `port30` to `port28`:

```
set_data_check 0.345 -rise_from port30 -to port28
report_timing -rise_from port30 -to port28
```

The following report is displayed:

```
Path 1: VIOLATED Data To Data Setup Check with Pin port30
Endpoint: port28 (v) checked with leading edge of 'PH1'
Beginpoint: U2/Q (v) triggered by leading edge of 'PH1'
Other End Arrival Time 0.465
- Data Check Setup 0.345
+ Phase Shift 0.000
= Required Time 0.120
- Arrival Time 0.309
= Slack Time -0.189
```

```
reset_data_check -rise_from port30 -to port28
report_timing -rise_from port30 -to port28
```

The following information is displayed:

```
No constrained timing paths with given description found.
Paths may be unconstrained (try '-unconstrained' option) or may not exist.
```

Related Commands

- `set_data_check`

reset_disable_clock_gating_check

```
reset_disable_clock_gating_check
pin_or_inst_list
```

Enables clock gating checks for the specified instances or pins.

Parameters

pin_or_inst_list	Specifies the list of disabled pins or instances that are to be enabled.

Related Information

- set_disable_clock_gating_check

reset_disable_timing

```
reset_disable_timing
[-help]
[-from pin_name -to pin_name]
obj_list
```

Restores timing arcs that were disabled using the `set_disable_timing` command.

Parameters

`-help`	Prints out the command usage.
`-from pin_name -to pin_name`	Specifies the arc(s) that are to be restored. Specify the `-from` and `-to` options together. You can specify the `-from` and `-to` parameters when the *`obj_list`* contains instance or library cell objects. Do not specify the `-from` and `-to` parameters when the *`obj_list`* includes instance pin or library cell pin objects. The `-from` and `-to` options are optional. If not specified, all cell arcs found in the instance or library cell objects are restored.
`obj_list`	Specifies the instance, instance pin, library arc, library cell, or library cell pin objects. If the *`obj_list`* includes instance pin or library cell pin objects, all cell arcs attached to the specified pin objects are restored. If the *`obj_list`* contains instance or library cell objects, and the `-from` and `-to` parameters are not specified, all arcs found in the specified instance or library cell objects are restored.

Examples

- The following commands set and reset the timing arcs and display the resulting reports:

```
set_disable_timing -from A -to Y [get_cells b1]
report_inactive_arcs

-------------------------------------------------------------
Flags :              const                propagated constant
                     snipped              loop snipped arcs
                     disable              set_disable_timing
                     disable_clock_gating set_disable_clock_gating
                     library_disable      set_disable_cell_timing
                     Missing_Phase        No Arc Phase Data
From       To       DisableType  ArcType         Reason
-------------------------------------------------------------
b1/A ^     b1/Y ^   disable      combinational   User Disable
b1/A v     b1/Y v   disable      combinational   User Disable
-------------------------------------------------------------
```

- The following commands disable the timing report header:

```
set_global timing_disable_report_header_info true
report_inactive_arcs

-------------------------------------------------------------------------
Flags :                  const                    propagated constant
                         snipped                  loop snipped arcs
                         disable                  set_disable_timing
                         disable_clock_gating     set_disable_clock_gating
                         library_disable          set_disable_cell_timing
                         Missing_Phase            No Arc Phase Data
From       To        DisableType    ArcType                 Reason
-------------------------------------------------------------------------
u8/CK ^    u8/D ^    const          hold_rising_clk_rise    u8/D = 1
u8/CK ^    u8/D v    const          hold_falling_clk_rise   u8/D = 1
u8/CK ^    u8/D ^    const          setup_rising_clk_rise   u8/D = 1
u8/CK ^    u8/D v    const          setup_falling_clk_rise  u8/D = 1
-------------------------------------------------------------------------
```

- The following commands disable the timing arcs from CK to Q for instance U8, then reset the assertions using the instance reference:

```
set_disable_timing -from CK -to Q [get_cells u8]
report_inactive_arcs
```

```
------------------------------------------------------------------
Flags :                 const                 propagated constant
                        snipped               loop snipped arcs
                        disable               set_disable_timing
                        disable_clock_gating  set_disable_clock_gating
                        library_disable       set_disable_cell_timing
                        Missing_Phase         No Arc Phase Data
From       To       DisableType     ArcType       Reason
------------------------------------------------------------------
u8/CK ^    u8/Q ^   disable         rising_edge   User Disable
u8/CK ^    u8/Q v   disable         rising_edge   User Disable
------------------------------------------------------------------
```

- The following commands disable the timing arcs from CK to Q for instance U8, then reset the assertions using the instance reference:

```
reset_disable_timing u8
report_inactive_arcs

------------------------------------------------------------------------
Flags :                   const                 propagated constant
                          snipped               loop snipped arcs
                          disable               set_disable_timing
                          disable_clock_gating  set_disable_clock_gating
                          library_disable       set_disable_cell_timing
                          Missing_Phase         No Arc Phase Data
From      To       DisableType     ArcType                   Reason
------------------------------------------------------------------------
u8/CK ^   u8/D ^   const           hold_rising_clk_rise      u8/D = 1
u8/CK ^   u8/D v   const           hold_falling_clk_rise     u8/D = 1
u8/CK ^   u8/D ^   const           setup_rising_clk_rise     u8/D = 1
u8/CK ^   u8/D v   const           setup_falling_clk_rise    u8/D = 1
------------------------------------------------------------------------
```

Related Commands

- set_disable_timing
- report_inactive_arcs

reset_drive

`reset_drive` *`port_list`*

Resets the drive previously asserted with the `set_drive` command on input ports. The `reset_drive` command removes assertions for all rise/fall, minimum/maximum combinations for which the assertion exists.

Parameters

`port_list`	Specifies a list of input or bidirectional ports for which the drive resistance is reset.

Related Information

- `set_drive`

reset_driving_cell

```
reset_driving_cell
[-rise]
[-fall]
[-min]
[-max]
port_list
```

Resets the driving cell assertions applied with the `set_driving_cell` command for the given rise/fall, minimum/maximum combination on ports.

If `-rise`, `-fall`, `-min`, or `-max` parameters are not specified, the `reset_driving_cell` command resets the assertion for all rise/fall, minimum/maximum combinations for which the assertion exists.

Parameters

`-rise`	Specifies the rising drive assertions.
`-fall`	Specifies the falling drive assertions.
`-min`	Specifies the early drive assertion.
`-max`	Specifies the late drive assertion.
port_list	Specifies a list of ports for which the driving cell assertions are reset.

Related Information

- `set_driving_cell`

reset_generated_clock

```
reset_generated_clock
[-help]
{-all | clock_list}
```

Removes generated clock assertions previously created using the `create_generated_clock` command.

To use the `reset_generated_clock` command you must specify the `set_interactive_constraint_modes` constraints for the applicable modes. You can also include these commands in the SDC constraint file - applicable only for the views that include that SDC.

Note: The generated clocks may originate from the library files dotlib - you can use `reset_generated_clock` from the command line, or add it to all the SDC files in use.

Parameters

`-help`	Prints out the command usage.
`-all`	Removes all the generated clocks in the design.
`clock_list`	Specifies the list of clocks (in collection format) for which the generated clock assertions are to be removed.

Examples

Here are some examples.

- The following command removes the generated clock assertions on all the generated clocks `RC/Q`:

```
> report_clocks

# The following report is displayed:

--------------------------------------------------------------------------
                              Clock Descriptions
--------------------------------------------------------------------------
                                                        Attributes
--------------------------------------------------------------------------
Clock Name  Source     Period     Lead     Trail    Generated   Propagated
--------------------------------------------------------------------------
CK1         CLK1       20.000     0.000    10.000     n           n
CK2         CLK1       10.000     0.000    5.000      n           n
GCLK        u8/Q       20.000     0.000    10.000     y           y
--------------------------------------------------------------------------

Generated-Clock Descriptions
-------------------------------------------------------------------------------
Name  Generated    Master      Master-clock  Invert  Freq.      Duty-Cycle Edges Edge-Shift
```

```
      Source(pin)  Source(pin)                        Multiplier
-------------------------------------------------------------------------------
GCLK  u8/Q         CK1         CLK1           n      1/2        -          -     -
-------------------------------------------------------------------------------

> set_interactive_constraint_modes [all_constraint_modes -active]
> reset_generated_clock [get_clocks GCLK]
> report_clocks

# The following report is displayed:
-------------------------------------------------------------------------
                         Clock Descriptions
-------------------------------------------------------------------------
                                                  Attributes
-------------------------------------------------------------------------
Clock Name   Source   Period   Lead    Trail    Generated     Propagated
-------------------------------------------------------------------------
CK1          CLK1    20.000   0.000   10.000    n             n
CK2          CLK1    10.000   0.000   5.000     n             n
GCLK         u8/Q    20.000   0.000   10.000    y             y
-------------------------------------------------------------------------
Generated-Clock Descriptions
-------------------------------------------------------------------------------
Name  Generated   Master      Master-clock Invert  Freq.      Duty-Cycle Edges Edge-Shift
      Source(pin) Source(pin)                      Multiplier
-------------------------------------------------------------------------------
```

Related Information

- create_generated_clock
- report_clocks
- set_interactive_constraint_modes

reset_ideal_latency

```
reset_ideal_latency
[-min]
[-max]
[-rise]
[-fall]
pin_or_port_list
```

Resets the ideal latency constraints, specified using the `set_ideal_latency` command, on instance pins or ports.

Parameters

`-fall`	Resets the ideal latency for the falling edge.
`-max`	Resets the ideal latency for the maximum corner.
`-min`	Resets the ideal latency for the minimum corner.
`pin_or_port_list`	Resets a list of objects - instance pins, ports, or nets.
`-rise`	Resets the ideal latency for the rising edge.

Related Information

- `set_ideal_latency`

reset_ideal_network

```
reset_ideal_network
obj_list
```

Resets the ideal network constraints, specified using the `set_ideal_network` command, on pins, ports, or nets.

Parameters

obj_list	Resets a list of objects - instance pins, ports, or nets.

Related Information

- `set_ideal_network`

reset_ideal_transition

```
reset_ideal_transition
[-min]
[-max]
[-rise]
[-fall]
pin_or_port_list
```

Resets the ideal transition constraint, specified using the `set_ideal_transition` command, on instance pins or ports.

Parameters

`-fall`	Resets the ideal transition for the falling edge.
`-max`	Resets the ideal transition for the maximum corner.
`-min`	Resets the ideal transition for the minimum corner.
`pin_or_port_list`	Resets a list of objects - instance pins, ports, or nets.
`-rise`	Resets the ideal transition for the rising edge.

Related Information

- `set_ideal_transition`

reset_input_delay

```
reset_input_delay
[-help]
[-clock clk_name]
[-clock_fall]
[-rise] [-fall]
[-max] [-min]
[-level_sensitive]
pin_or_port_list
```

Resets previously specified input delay assertions.

Parameters

`-help`	Prints out the command usage.
`-clock` *clock_name*	Specifies the name of the clock. *Default*: The asynchronous (`@`) clock. If you specify multiple clocks, an error occurs and the constraint is ignored. When you specify the `-clock` parameter, the constraints specified for a pin with matching `-clock` parameter are reset. If you do not specify the `-clock` parameter, all the constraints applied on the specified pin, with or without the `-clock` parameter specification, are reset.
`-clock_fall`	Removes the assertions for the delay that is relative to the falling edge of the clock. You must specify the `-clock` parameter with this parameter. Default: Removes the delay assertions relative to the falling edge.
`-fall`	Resets the input delay for the falling edge at the input port. If both the `-rise` and `-fall` options are not specified, then the input delay is reset on both the edges.
`-level_sensitive`	Resets input delay that is launched by a positive level sensitive latch. When you specify the `-level_sensitive` parameter, the command resets all the `set_input_delay` constraints specified with the `-level_sensitive` parameter. If `-level_sensitive` parameter is not specified, the constraints specified for the same pin, with or without the `-level_sensitive` parameter, are reset. **Note:** You must specify the `-clock` parameter with this parameter.
`-max`	Removes the delay assertions that refer to the setup analysis. If you do not specify the `-max` parameter, the input delay assertions for both setup and hold analysis are removed.

`-min`	Removes the delay assertions that refer to the hold analysis. If you do not specify the `-min` parameter, the input delay assertions for both setup and hold analysis are removed.
`pin_or_port_list`	Specifies a single or multiple pins or ports for which the input delay is reset. To specify multiple pins, enclose the list with curly braces (`{}`) and separate the pin information with white space.
`-rise`	Resets the input delay for the rising edge at the input port. If both `-rise` and `-fall` options are omitted, the input delay is reset on both the edges.

Related Information

- `set_input_delay`

reset_load

```
reset_load
port_net_list
```

Resets the existing `set_load` assertions specified on nets or ports.

Parameters

port_net_list	Specifies a list of nets or ports in the top cell for which assertions are reset.

Related Information

- `set_load`

reset_max_capacitance

```
reset_max_capacitance
obj_list
```

Resets maximum capacitance limit specified (using the the `set_max_capacitance` command) on the objects in the object list. The valid objects in the object list are primary ports, clocks, and current design.

Parameters

obj_list	Specifies the list of ports, modules, and/or clock waveforms for which assertions are reset.

Related Information

- `set_max_capacitance`

reset_max_fanout

```
reset_max_fanout
obj_list
```

Resets maximum fanout limit specified (using the `set_max_fanout` command) on the objects in the object list. The valid objects in the object list are input and bidirectional ports and current design.

Parameters

obj_list	Specifies the list of ports or modules for which assertions are reset.

Related Information

- `set_max_fanout`

reset_max_time_borrow

```
reset_max_time_borrow
obj_list
```

Resets the specified maximum time borrow limit of the specified objects.

Parameters

obj_list	Specifies the list of pins, instances, or clocks for which the time borrow limit is reset.

Related Information

- set_max_time_borrow

reset_max_transition

```
reset_max_transition
obj_list
```

Resets the maximum slew (transition) limit specified using the `set_max_transition` command (from the SDC or the command line) on the objects in the given object list. The valid objects in the object list are primary ports, clocks, and current design.

The maximum transition limit that is set by the user overrides the dotlib value so that sharper slews are implemented on the timing sensitive nets, such as clocks. A violation due to the `set_max_transition` setting does not impact the timing accuracy.

Note: For command-line usage of this command, you need to use `set_interactive_constraint_modes`, for example, `set_interactive_constraint_modes [all_constraint_modes -active]`

Parameters

obj_list	Specifies the list (or collection) of ports, modules, and/or clock waveforms for which assertions are to be reset. For example, `reset_max_transition [get_ports *]`

Related Information

- `set_max_transition`

reset_min_capacitance

```
reset_min_capacitance
obj_list
```

Resets minimum capacitance limit specified (using the the `set_min_capacitance` command) on objects in the object list. The valid objects in the object list are primary ports, clocks, and the current design.

Parameters

obj_list	Specifies the list of ports, modules, and/or clock waveforms for which assertions are reset.

Related Information

- `set_min_capacitance`

reset_min_fanout

```
reset_min_fanout
obj_list
```

Resets minimum fanout limit specified (using the `set_min_fanout` command) on the objects in the object list. The valid objects in the object list are input and bidirectional ports and the current design.

Parameters

obj_list	Specifies the list of ports or modules for which assertions are reset.

Related Information

- `set_min_fanout`

reset_min_pulse_width

```
reset_min_pulse_width
[-help]
obj_list
[-all]
[-high]
[-low]
[-waveform_aware_type {absolute | source_width_ratio | all}]
```

Resets minimum pulse width constraint on the specified input ports of the top cell and the specified modules.

Parameters

`-help`	Prints out the command usage.
`-all`	Resets the regular and waveform-aware pulse width assertions.
`-low`	Resets the minimum pulse width checks that are applied to low signal levels only.
`-high`	Resets the minimum pulse width checks that are applied to high signal levels only.
obj_list	Specifies a list of pins or clock waveforms for which the minimum pulse width is to be reset. If a clock waveform is specified, all the pins driven by the clock are affected.
`-waveform_aware_type {absolute | source_width_ratio | all}`	Resets the regular and waveform-aware pulse width assertions.

Related Information

- `set_min_pulse_width`

reset_min_transition

```
reset_min_transition
obj_list
```

Resets minimum slew (transition) limit specified (using the `set_min_transition` command) on the objects in the object list. The valid objects in the object list are primary ports, clocks, and current design.

Parameters

obj_list	Specifies the list of ports, modules, and/or clock waveforms for which assertions are reset.

Related Information

- `set_min_transition`

reset_mode

```
reset_mode
[-type cell]
mode_list
inst_list
```

Resets the Liberty timing library modes for a specified instance.

Library modes are set using the set_mode command. By default, all library modes in the timing library are active. If you specify one or more library modes using the `set_mode` command, only the timing arcs consistent with the specified mode(s) are active for the instance. All other library modes become inactive. When you reset the library mode for the instance, all library modes become active again.

Parameters

inst_list	Specifies the instances to which to reset the library modes.
mode_list	Specifies the name of the library modes to reset.
-type cell	Resets the library mode at the instance level. **Note:** Setting the library mode at the instance level is the only form of `set_mode` that is currently supported. This parameter exists to support constraints coming from external sources.

Examples

- The following command specifies that only the timing arcs consistent with the `test_A` library mode are active for instances of b1:
  The test_A library mode is made inactive.

  ```
  set_mode test_mode [get_cells BUF]
  ```

  The following command reports the status of all library modes associated with BUF 'b1' . It shows that test_A is active and test_B is inactive.

  ```
  report_mode b1
  ------------------------------
  Mode Group test_mode of b1
  ------------------------------
  Mode     Status    Condition
  Name
  ------------------------------
  test_B   Inactive  !(Y)
  test_A   ACTIVE    Y
  ------------------------------
  ```

  The following command resets the library mode for instances of b1 :

  ```
  reset_mode test_A b1
  ```

  If you generate a report of the library mode status for b1, it shows that both modes are now active:

  ```
  report_mode b1
  ------------------------------
  Mode Group test_mode of b1
  ------------------------------
  Mode    Status   Condition
  Name
  ------------------------------
  test_B  ACTIVE    !(Y)
  test_A  ACTIVE    Y
  ------------------------------
  ```

Related Commands

- set_mode
- report_mode

reset_output_delay

```
reset_output_delay
[-help]
port_or_pin_list
[-clock clock_name]
[-clock_fall]
[-fall]
[-level_sensitive]
[-max]
[-min]
[-rise]
```

Resets previously specified output delay assertions.

Parameters

`-help`	Prints out the command usage.
`-clock` *`clock_name`*	Specifies the name of the clock. *Default*: The asynchronous (`@`) clock. If you specify multiple clocks, an error occurs and the constraint is ignored. When you specify the `-clock` parameter, the constraints specified for a pin with matching `-clock` parameter are reset. If you do not specify the `-clock` parameter, all the constraints applied on the specified pin, with or without the `-clock` parameter specification, are reset.
`-clock_fall`	Removes the assertions for the delay that is relative to the falling edge of the clock. You must specify the -clock parameter with this parameter. *Default:* Removes the delay assertions relative to the rising edge.
`-fall`	Resets the output delay for the falling edge at the output port. If both `-rise` and `-fall` options are omitted, the output delay is reset on both the edges.
`-level_sensitive`	Resets output delay that is launched by a positive level sensitive latch. When you specify the `-level_sensitive` parameter, the command resets all the `set_output_delay` constraints specified with the `-level_sensitive` parameter. If `-level_sensitive` parameter is not specified, the constraints specified for the same pin, with or without the `-level_sensitive` parameter, are reset. **Note:** You must specify the `-clock` parameter with this parameter.
`-max`	Removes the delay assertions that refer to the setup analysis. If you do not specify the `-max` parameter, the output delay assertions for both setup and hold analysis are removed.
`-min`	Removes the delay assertions that refer to the hold analysis. If you do not specify the `-min` parameter, the output delay assertions for both setup and hold analysis are removed.
`pin_or_port_list`	Specifies a single or multiple pins or ports for which the output delay is reset. To specify multiple pins, enclose the list with curly braces (`{}`) and separate the pin information with white space.
`-rise`	Resets the output delay for the rising edge at the output port. If both `-rise` and `-fall` options are omitted, the output delay is reset on both the edges.

Examples

- The following example sets the output delays for the output port Z. The output signal is required at Z at 2 units before the rising edge of clock Clk1, and at 2.5 units before falling edge of clock Clk2:

```
set_output_delay 2 Z -clock Clk1
set_output_delay 2.5 Z -clock Clk2 -clock_fall -add_delay
```

- The following command will reset the output delay for the rising edge of clock Clk2:

```
reset_output_delay Z -clock Clk2

report_timing -to_rise Z -clock_to Clk2
#################################################################
# Generated by: Cadence Innovus
# OS: Linux x86_64(Host ID noi-rp)
# Generated on: Thu Apr 18 09:50:52 2019
# Design: top
# Command: report_timing -to_rise Z -clock_to Clk2
#################################################################
No constrained timing paths with given description found.
Paths may be unconstrained (try '-unconstrained' option) or may not exist.

report_timing -to_fall Z -clock_to Clk2
#################################################################
Path 1: VIOLATED Late External Delay Assertion
Endpoint: Z (v) checked with trailing edge of 'Clk2'
Beginpoint: dff3/Q (v) triggered by leading edge of 'Clk'
Path Groups: {vclk2}
Other End Arrival Time 2.000
- External Delay 2.500
+ Phase Shift 0.000
= Required Time -0.500
- Arrival Time 0.165
= Slack Time -0.665
Clock Rise Edge 0.000
+ Clock Network Latency (Prop) 0.052
= Beginpoint Arrival Time 0.052
-------------------------------------------------------------
Instance Arc Cell Delay Arrival Required
Time Time
-------------------------------------------------------------
dff3 CK ^ - - 0.052 -0.613
dff3 CK ^ -> Q v DFFHQX1 0.113 0.165 -0.500
- Z v - 0.000 0.165 -0.500
-------------------------------------------------------------
```

- The following command resets the output delay for clock Clk2 with respect to the falling edge:
  ```
  reset_output_delay Z -clock Clk2 -clock_fall
  ```
- The following command will reset all the output delays with respect to all the edges for both the clocks - Clk1 and Clk2:
  ```
  reset_output_delay Z
  ```

Related Commands

- set_output_delay

reset_path_exception

```
reset_path_exception
[-exact]
[-type {false_path | multicycle | path_delay}]
[-rise]
[-fall]
[{-from | -rise_from | -fall_from} from_list]
[{-through | -rise_through | -fall_through} through_list]
[{-to | -rise_to | -fall_to} to_list]
[-setup | -hold]
[-all]
```

Removes any previously set path exceptions (or path exception of the specified type) for the given paths. To remove all path exceptions, use the `-all` option with the `reset_path_exception` command.

Parameters

`-all`	Removes all path exceptions. This option cannot be used with the `-from`, `-through and -to` options.
`-exact`	Forces exact matching of the constraints. When this parameter is specified, the reset settings (-from/-through/-to) match with the original constrained path settings.
`-fall`	Applies the assertion at the falling edge of the signal on the path endpoints.
`{-from | -rise_from | -fall_from} from_list`	Specifies pins at the start of the path exceptions that are to be reset.
`-rise`	Applies the assertion at the rising edge of the signal on the path endpoints.
`{-through | -rise_through | -fall_through} through_list`	Specifies the pins that the path to be reset goes through.
`{-to | -rise_to | -fall_to} to_list`	Specifies pins at the end of the path exceptions that are to be reset.
`-setup | -hold`	Resets only the setup (or hold) paths. The `-setup`/`-hold` parameters should be used to reset maximum or minimum delay that is set using the `set_max_delay` and `set_min_delay` commands. *Default*: Both `-setup` and `-hold` are reset.
`-type {false_path | multicycle | path_delay}`	Removes only path exceptions of the specified type. *Default*: All path exception assertions are reset.

Examples

- In the example below the rising path from R1 to R2 has three different path exceptions asserted on it - they are listed in priority from lowest to highest. As shown in the reports below, the false path is the highest precedence and so no path is reported.

```
set_multicycle_path 2 -from R1 -rise_to R2
set_max_delay 3 -from R1 -rise_to R2
set_false_path -from R1 -rise_to R2

report_timing -from R1 -rise_to R2
```

  # The following information is displayed:

```
No constrained timing paths with given description found.
Paths may be unconstrained (try '-unconstrained' option) or may not exist.
```

```
report_path_exceptions
```

  # The following report is displayed:

```
+---------------------------------+
| From  | To     | Late           |
|-------+--------+----------------|
| R1/CK | ^ R2/D | delay 3 ignored |
| R1/CK | ^ R2/D | add 1 ignored   |
| R1/CK | ^ R2/D | false           |
+---------------------------------+
```

- In this example, the false path exception is removed from the path. This causes the max path delay assertion to become active - as it is the next highest precedence.

```
reset_path_exception -rise_to R2/D -type false_path

report_path_exceptions
```

  # The following report is displayed:

```
+--------------------------------+
| From  | To     | Late          |
|-------+--------+---------------|
| R1/CK | ^ R2/D | delay 3       |
| R1/CK | ^ R2/D | add 1 ignored |
+--------------------------------+
```

```
report_timing -from R1 -rise_to R2/D
```

The following report is displayed:

```
Path 1: MET Setup Check with Pin R2/CK
Endpoint: R2/D (^) checked with leading edge of 'PH1'
Beginpoint: R1/Q (^) triggered by leading edge of 'PH1'
Other End Arrival Time 0.000
- Setup 0.077
+ Path Delay 3.000 <---------
= Required Time 2.923
- Arrival Time 0.238
= Slack Time 2.685
Clock Rise Edge 0.000
= Beginpoint Arrival Time 0.000
+-------------------------------------------------------------------+
| Instance | Arc         | Cell      | Delay | Arrival | Required |
|          |             |           |       | Time    | Time     |
|----------+-------------+---------+-------+---------+-----------|
|          | clk ^       | | | 0.000 | 2.685 |
| R1       | CK ^ -> Q ^ | DFFHQX1   | 0.170 | 0.170   | 2.855 |
| U2       | A ^ -> Y ^  | BUFX1     | 0.068 | 0.238   | 2.923 |
| R2       | D ^         | DFFHQX1   | 0.000 | 0.238   | 2.923 |
+-------------------------------------------------------------------+
```

- In this example, the `set_max_delay` exception is removed, leaving only the `set_multicycle_path` exception active.

```
reset_path_exception -rise_to R2/D -type path_delay

report_path_exceptions
```

The following report is displayed:

```
+-------------------------+
| From  | To     | Late  |
|-------+--------+-------|
| R1/CK | ^ R2/D | add 1 |
+-------------------------+

report_timing -from R1 -rise_to R2/D
```

The following report is displayed:

```
Path 1: MET Setup Check with Pin R2/CK
Endpoint: R2/D (^) checked with leading edge of 'PH1'
Beginpoint: R1/Q (^) triggered by leading edge of 'PH1'
Other End Arrival Time 0.000
- Setup 0.077
+ Phase Shift 10.000
+ Cycle Adjustment 10.000 <----- Multi-Cycle timing
= Required Time 19.923
- Arrival Time 0.238
= Slack Time 19.685
Clock Rise Edge 0.000
= Beginpoint Arrival Time 0.000
+-------------------------------------------------------------+
| Instance | Arc          | Cell    | Delay | Arrival | Required |
|          |              |         |       | Time    | Time     |
|----------+--------------+---------+-------+--------------------|
|          | clk ^        |         |       | 0.000 | 19.685     |
| R1       | CK ^ -> Q ^  | DFFHQX1 | 0.170 | 0.170 | 19.855     |
| U2       | A ^ -> Y ^   | BUFX1   | 0.068 | 0.238 | 19.923     |
| R2       | D ^          | DFFHQX1 | 0.000 | 0.238 | 19.923     |
+-------------------------------------------------------------+
```

- In this example, the multi-cycle path assertion is removed, and the analysis reverts back to single-cycle timing

```
reset_path_exception -rise_to R2/D -type multicycle
report_path_exceptions
```

None reported

```
report_timing -from R1 -rise_to R2/D
```

The following report is displayed:

```
Path 1: MET Setup Check with Pin R2/CK
Endpoint: R2/D (^) checked with leading edge of 'PH1'
Beginpoint: R1/Q (^) triggered by leading edge of 'PH1'
Other End Arrival Time 0.000
- Setup 0.077
+ Phase Shift 10.000 <--- Single-cycle timing
= Required Time 9.923
```

```
- Arrival Time 0.238
= Slack Time 9.685
Clock Rise Edge 0.000
= Beginpoint Arrival Time 0.000
+-------------------------------------------------------------+
| Instance | Arc          | Cell    | Delay | Arrival | Required |
|          |              |         |       | Time    | Time     |
|----------+--------------+---------+-------+---------+----------|
|          | clk ^        |         |       | 0.000   | 9.685    |
| R1       | CK ^ -> Q ^  | DFFHQX1 | 0.170 | 0.170   | 9.855    |
| U2       | A ^ -> Y ^   | BUFX1   | 0.068 | 0.238   | 9.923    |
| R2       | D ^          | DFFHQX1 | 0.000 | 0.238   | 9.923    |
+-------------------------------------------------------------
```

- Consider the following `set_false_path` constraint:

  ```
  set_false_path -from {u1/Y u2/Y} -to {u3/Y}
  ```

  When you specify `reset_path_exception -from {u1/Y}` command without the `-exact` parameter, the command outputs the following:

  ```
  set_false_path -from {u2/Y} -to {u3/Y}
  ```

  The command `reset_path_exception -from {u1/Y} -exact` outputs the following:

  ```
  set_false_path -from {u1/Y u2/Y} -to {u3/Y}
  ```

  In this case (when `-exact` parameter is used), the path exception is not reset because differential matching is not allowed.

Related Information

- `set_false_path`

reset_path_group

```
reset_path_group
[-name group_name]
[-all]
```

Removes the specified path groups or completely removes all the groups.

Parameters

`-all`	Removes all path groups.
`-name group_name`	Removes the named group completely.

Example

- The following command removes the group named GRPA:
  ```
  reset_path_group -name GRPA
  ```

Related Information

- `group_path`

reset_pll_timing

```
reset_pll_timing
[-help]
<inst>
-pll_output_clock <clock name or ID>
-pll_output_pin <output pin name>
```

Resets the phase locked loop (PLL) that is used to adjust the delay on the PLL output clock waveform.

Parameters

`-help`	Prints out the command usage.
<*inst*>	Specifies the instance on which we will search reference pin, feedback pin, or output pin.
`-pll_output_clock <clock name or ID>`	Specifies the PLL output clock.
`-pll_output_pin <output pin name>`	Specifies the PLL output pin.

Related Commands

- `set_pll_timing`

reset_propagated_clock

```
reset_propagated_clock
[-help]
pin_or_clock_list
```

Removes the propagated clock assertion for the specified clock waveforms or pins.

The ideal or propagated status of clock endpoints in the fanout of the specified pin/clock list is determined by the remaining clock latency propagated clock assertions in the clock network.

When new analysis views are used, the new corners do not have the source latency updated as per the clock propagation settings, so a reset and re-calculation is required.

Note: You must specify `set_interactive_constraint_modes` constraint prior to running the `reset_propagated_clock` command.

Parameters

`-help`	Prints out the command usage.
`pin_or_clock_list`	Specifies a list of pins or clocks for which you want to reset the clock propagation mode to ideal mode. The *`pin_or_clock_list`* parameter can be a collection, however, no hierarchical pins are allowed in the list.

Examples

- The following command sets the `CLK1` clock waveform to `propagated` mode:

  ```
  set_propagated_clock CLK1
  ```

- The following command resets the `CLK1` clock waveform back to `ideal` mode:

  ```
  reset_propagated_clock CLK1
  ```

- The following commands set the `CLK1` clock to `propagated` mode and then reports them:

  ```
  set_propagated_clock [get_clocks CLK1]
  report_clocks
  ```

  The following report is displayed:

  ```
  ------------------------------------------------------------------------
                            Clock Descriptions
  ------------------------------------------------------------------------
                                                        Attributes
  ------------------------------------------------------------------------
  Clock Name      Source   Period    Lead    Trail    Generated   Propagated
  ------------------------------------------------------------------------
  CLK1            CLK1      20.000   0.000    10.000    n           y
  CLK2            CLK2      25.000   0.000    12.500    n           y
  CLK3            CLK3      10.000   0.000    6.000     n           y
  GCLK3           u8/Q      20.000   0.000    10.000    y           y
  ------------------------------------------------------------------------
  Generated-Clock Descriptions
  ------------------------------------------------------------------------
  Name  Generated     Master     Master-clock  Invert  Freq.       Duty-Cycle  Edges  Edge-Shift
        Source(pin)   Source(pin)                      Multiplier
  ------------------------------------------------------------------------
  GCLK3  u8/Q         CLK3       CLK3          n       1/2          -           -     -
  ------------------------------------------------------------------------
  ```

- The following commands reset the `CLK1` clock to `ideal` mode and then reports them:

  ```
  reset_propagated_clock [get_clocks CLK1]
  report_clocks
  ```

The following report is displayed:

```
--------------------------------------------------------------------------------
                            Clock Descriptions
--------------------------------------------------------------------------------
                                                          Attributes
--------------------------------------------------------------------------------
Clock Name    Source   Period   Lead    Trail    Generated      Propagated
--------------------------------------------------------------------------------
CLK1          CLK1     20.000   0.000   10.000   n              n
CLK2          CLK2     25.000   0.000   12.500   n              y
CLK3          CLK3     10.000   0.000   6.000    n              y
GCLK3         u8/Q     20.000   0.000   10.000   y              y
--------------------------------------------------------------------------------
Generated-Clock Descriptions
--------------------------------------------------------------------------------
Name    Generated    Master       Master-clock   Invert  Freq.       Duty-Cycle   Edges   Edge-
Shift
        Source(pin)  Source(pin)                         Multiplier
--------------------------------------------------------------------------------
GCLK3   u8/Q         CLK3         CLK3            n      1/2         -            -       -
--------------------------------------------------------------------------------
```

Related Commands

- create_clock
- get_propagated_clock
- report_clocks
- set_clock_latency
- set_propagated_clock
- set_interactive_constraint_modes

reset_property

```
reset_property
[-obj_type {object_type}]
{obj_list}
property_name
```

Resets user-defined properties on objects set using the `set_property` command.

Parameters

obj_list	Specifies a list of objects for which the property is to be reset.
-obj_type {*object_type*}	Specifies the type of object on which the property is to be reset.
property_name	Specifies the name of the property for which the user-defined property is to be reset.

Examples

- The following command resets user-defined property `slk` from pins `C1/A`:

  ```
  reset_property [get_pins C1/A] slk
  ```

Related Commands

- `get_property`
- `set_property`
- `define_property`

reset_pulse_clock_max_transition

```
reset_pulse_clock_max_transition
obj_list
[-transitive_fanout]
[-rise]
[-fall]
```

Resets the maximum pulse clock transition constraint specified (using the `set_pulse_clock_max_transition` command) on the pulse generator cells, pulse generator library cells, clocks, and/or current design.

Parameters

`-fall`	Resets transition limit value for falling transition.
`obj_list`	Specifies a list of objects - pulse generator cells, pulse generator library cells, clocks, or current design - for which the transition value is to be reset.
`-rise`	Resets transition limit value for rising transition.
`-transitive_fanout`	Specifies the transition limit value for transitive fanout.

Related Information

- `set_pulse_clock_max_transition`

reset_pulse_clock_max_width

```
reset_pulse_clock_max_width
obj_list
[-transitive_fanout]
```

Resets the maximum pulse clock width constraint specified (using the `set_pulse_clock_max_width` command) on the pulse generator cells, pulse generator library cells, clocks, and/or current design.

Parameters

obj_list	Specifies a list of objects - pulse generator cells, pulse generator library cells, clocks, or current design - for which the maximum pulse clock width is to be reset.
[-transitive_fanout]	Specifies the pulse width value for transitive fanout.

Related Information

- `set_pulse_clock_max_width`

reset_pulse_clock_min_transition

```
reset_pulse_clock_min_transition
obj_list
[-transitive_fanout]
[-rise]
[-fall]
```

Resets the minimum pulse clock transition constraint specified (using the `set_pulse_clock_min_transition` command) on the pulse generator cells, pulse generator library cells, clocks, and/or current design.

Parameters

`-fall`	Resets transition limit value for falling transition.
`obj_list`	Specifies a list of objects - pulse generator cells, pulse generator library cells, clocks, or current design - for which the minimum pulse clock transition value is to be reset.
`-rise`	Resets transition limit value for rising transition.
`-transitive_fanout`	Specifies the transition limit value for transitive fanout.

Related Information

- `set_pulse_clock_min_transition`

reset_pulse_clock_min_width

```
reset_pulse_clock_min_width
obj_list
[-transitive_fanout]
```

Resets the minimum pulse clock width constraint specified (using the `set_pulse_clock_min_width` command) on the pulse generator cells, pulse generator library cells, clocks, and/or current design.

Parameters

obj_list	Specifies a list of objects - pulse generator cells, pulse generator library cells, clocks, or current design - for which the minimum pulse clock width is to be reset.
[-transitive_fanout]	Specifies the pulse width value for transitive fanout.

Related Information

- `set_pulse_clock_min_width`

reset_resistance

```
reset_resistance
net_list
```

Resets the existing `set_resistance` assertions specified on nets.

Parameters

net_list	Specifies a list of nets for which the wire resistance is reset.

Related Information

- `set_resistance`

set_active_clocks

```
set_active_clocks
[-help]
active_clock_list | all_clocks
```

Defines a list of active clocks in the design.

When the `set_active_clocks` command is specified, only the specified clocks are made active while others remain inactive. Any clock created after using the `set_active_clocks` command is active by default. The following commands are impacted:

- `report_clocks` command output displays a new column showing active or inactive clocks in the design.
- `get_clocks` command will report all clocks including inactive clocks.
- `write_timing_windows` command will not display windows for inactive clocks in the output report.
- `get_property` command allows you to query on active clocks using the `is_clock_active` property.

Parameters

`-help`	Prints out the command usage.
`active_clock_list` | *`all_clocks`*	Specifies a list of active clocks.

Examples

- The following command makes the specified clocks active while others remain inactive (only `Iclk1` is active, while `Iclk2` is inactive):

```
set_active_clocks Iclk1
```

Related Commands

- `report_clocks`
- `get_clocks`
- `get_property`
- `write_timing_windows`

set_annotated_check

```
set_annotated_check
[-help]
value
[-clock clock_check]
[-cond sdf_condtion]
[-fall]
-from from_pins
[-incremental]
[-max]
[-min]
[-rise]
-to to_pins
{-setup | -hold | -recovery | -removal | -nochange_high | -nochange_low}
```

Annotates the setup, hold, recovery, or removal timing check value between two or more pins of a cell in the current design.

Parameters

`-help`	Prints out the command usage.
`value`	Specifies the timing check value.
`-clock` *`clock_check`*	Specifies clock rising or falling. Default: Sets checks for both clock rise and fall. Use the *`clock_check`* argument to specify rise or fall.
`-cond` *`sdf_condition`*	Specifies the condition only if the library has a condition attached to the specified timing check arc. The *`sdf_expression`* must match the syntax of the condition specified in the library.
`-fall`	Specifies the data fall transition for the timing check. Default: Sets both `-rise` and `-fall`.
`-from` *`from_pins`*	Specifies start points of the timing arcs for which checks are annotated. Use the *`from_pins`* argument to specify a list of leaf cell clock pins.
`-incremental`	Sets the incremental checks.
`-max`	Specifies the maximum timing check for both data rise and data fall transitions. Use this option only if the design uses minimum and maximum (`min-max`) operating conditions.
`-min`	Specifies the minimum timing check for both data rise and data fall transitions. Use this option only if the design uses minimum and maximum (`min-max`) operating conditions.
`-rise`	Specifies the data rise transition for the timing check. Default: Sets both rise and fall.
`-setup | -hold | -recovery | -removal | -nochange_high | -nochange_low`	Specifies the timing check type. Make sure there is a corresponding timing arc between the *`from_pins`* and the *`to_pins`* arguments.
`-to` *`to_pins`*	Specifies endpoints of the timing arcs for which checks are annotated. Use the *`to_pins`* argument to specify a list of leaf cell clock pins.

Examples

- The following commands annotates a `nochange check` between data low and clock low. The nochange margin before the clock is `1.3` and the nochange margin after the clock is `2.4`:

```
set_annotated_check -nochange_low 1.3 -clock fall -fall -from {I1/CP} -to {I1/EN}
set_annotated_check -nochange_low 2.4 -clock fall -rise -from {I1/CP} -to {I1/EN}
```

- The following command annotates a 2.0 setup time between the CP clock pin and the A data pin of the

<*instance_name*> instance:

```
set_annotated_check -setup 2.0 -from <instance_name>/CP -to <instance_name>/A
```

Related Commands

- reset_annotated_check
- set_annotated_delay

set_annotated_delay

```
set_annotated_delay
[-help]
delay_value
[-cond expression]
[-delta_only]
[-fall]
[-increment]
[-max]
[-min]
[-rise]
{-net | -cell }
{[-from from_pins] [-to to_pins]}
```

Annotates delay to timing arcs. If you specify the `set_annotated_delay` command without any parameters, the delay value specified is assumed to include both the base delay and any delta (cross-talk) delay.

Parameters

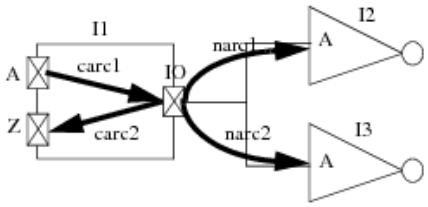
`-help`	Prints out the command usage.
`-cond expression`	Specifies the sdf condition of the selected timing arcs. *Default*: The delay applies to all the timing arcs selected using the `-from` and the `-to` options. The expression is ignored for net arcs. **Note:** You cannot specify the `-cond` parameter with `-net`. If specified, an error is displayed.
value	Specifies the delay value to be annotated to the selected timing arcs.
`-delta_only`	Populates the incremental delay slot with the delay value, so that you can differentiate between the base net delay and a delay offset that is due to crosstalk. The software reports the incremental (or delta) delay value separately in the timing report using `report_timing -format incr_delay`, and maintains the separate value for CPPR calculations. If the base delay was previously calculated or annotated from the SDF, the `-delta_only` parameter adds the specified delay to the existing base delay. **Note:** This parameter only can be used with the `-net` parameter; you cannot specify it with the `-cell` parameter.
`-from from_pins`	Specifies timing arcs to which the delay applies. The selected timing arcs are those that have their source pins specified using this option and the sink pins specified using the `-to` option. If this option is not specified, only timing arcs whose sink pins are specified using the `-to` option are selected. **Note:** You cannot specify hierarchical pins. If specified, an error message is displayed.
`-increment`	Increments the specified delay value on top of the existing base delay value.

`-min | -max`	Specifies the annotated delay for a particular operating corner. You can apply the `-min` parameter to the minimum operating corner and the `-max` parameter to the maximum operating corner.
`-net | -cell`	Specifies whether the delay should be annotated to cell arcs or net arcs.
`-rise/-fall`	Specifies the transition to which the delay applies. For cell arc, this is the output transition. *Default*: The delay applies to both transitions.
`-to` *to_pins*	Specifies timing arcs to which the delay applies. The selected timing arcs are those that have their sink pins specified using this option and the source pins specified using the `-from` option. If this option is not specified, only timing arcs whose source pins are specified using the `-from` option are selected. **Note:** You cannot specify hierarchical pins. If specified, an error message is displayed.

Examples

- The following example shows the usage of `-cell` and `-net` options:

  Figure: Example Figure to Show How to Use the -cell and -net Options

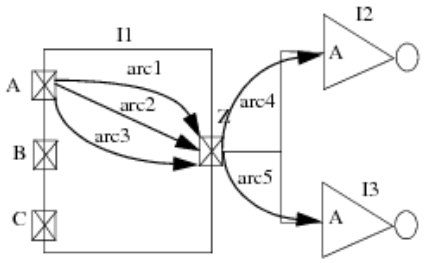
  

  - Use the `-cell` option to select the cell arc `carc2` only:
    set_annotated_delay 3.0 -cell -from {I1/IO}
  - Use the `-net` arc option to select net arcs `narc1` and `narc2` only:
    set_annotated_delay 3.0 -net -from {I1/IO}

- This TLF description is used for the following examples:

```
CELL (NAND3 . . . . // arc 1 Path( A => Z 10 01 DELAY(ioDelayRiseModel0) SLEW(SlopeRiseModel0) )
Path( A => Z 01 10 DELAY(ioDelayFallModel0) SLEW(SlopeFallModel0) ) // arc2 Path( A => Z 10 01
COND(B & C) SDF_COND(B & C) DELAY(ioDelayRiseModel1) SLEW(SlopeRiseModel1) ) Path( A => Z 01 10
COND(B & C) SDF_COND(B & C) DELAY(ioDelayFallModel1) SLEW(SlopeFallModel1) ) arc3 Path( A => Z 10
01 COND(~B | ~C) SDF_COND(!B | !C) DELAY(ioDelayRiseModel2) SLEW(SlopeRiseModel2) ) Path( A => Z 01
10 COND(~B | ~C) SDF_COND(!B | !C) DELAY(ioDelayFallModel2) SLEW(SlopeFallModel2) ) . . . }
```

  Figure: Example Figure to Show How to Use the -from and -to Options



- The following command annotates a delay of `3.0` to arcs `arc1`, `arc2`, and `arc3` for the rise and fall transition at the pin I1/Z, for the two corners `{min typ max}`: `set_annotated_delay 3.0 -from A -to Z -cell`
- The following command annotates a delay of 2.5 to net arc `arc4` for both the rising and the falling transition and for the two corners `{min typ max}`: `set_annotated_delay 2.5 -from I1/Z -to I2/A -net`
- The following command annotates a delay of 1.5 to `arc2` for the rising transition and the min corner:
  `set_annotated_delay 1.5 -from I1/A -to I1/Z -rise -min -cond "B & C" -cell`
- The following commands annotate a delay of 3.0 to `arc1` and `arc2`, and a delay of 2.4 to the `arc3`:
  `set_annotated_delay 3.0 -from I1/A -to I1/Z -cell set_annotated_delay 2.4 -from I1/A -to I1/Z -cond {!B | !C} -cell`
- The following commands first annotate a delay of 3.0 to `arc1`, `arc2` and `arc3`, then overwrite the value set by the first `set_annotated_delay` command: `set_annotated_delay 2.4 -from I1/A -to I1/Z -cond {!B | !C} -cell set_annotated_delay 3.0 -from I1/A -to I1/Z -cell`

Related Commands

- `set_annotated_check`

set_annotated_transition

```
set_annotated_transition
[-rise | -fall]
[-min | -max]
value
pin_list
```

Sets the transition time to be annotated on specified ports or pins in the current design. The transition time that you specify using this command overrides the transition time calculated internally within the software. You can use this command to set the transition time of internal cell instance pins.

Parameters

`-min | -max`	Specifies whether the transition value is set for minimum or maximum transition time. *Default:* Transition value is set for both transition times.
pin_list	Specifies a list of pins or ports to be annotated with the transition time that you specify using the *value* parameter.
`-rise | -fall`	Specifies whether the transition value is set for rising or falling transition time. *Default*: Transition value is set for both transitions.
value	Specifies the transition time to be annotated to the selected pins or ports.

Examples

- The following command annotates a transition time value of 0.6 on port A:
  `set_annotated_transition 0.6 [get_ports A]`
- The following command annotates a transition time value of 0.7 on pin I2/A: `set_annotated_transition 0.7 [get_pins I2/A]`
- The following command annotates a transition value of 0.7 on pin I2/A for rising transition time:
  `set_annotated_transition -rise 0.7 [get_pins I2/A]`
- The following command annotates a transition value of 0.7 on pin I2/A for rising transition time and minimum pvt corner: `set_annotated_transition -rise -min 0.7 [get_pins I2/A]`

Related Commands

- `set_input_transition`

set_aocv_interface_path_offset

```
set_aocv_interface_path_offset
derate_offset_value
[lib_cell_list]
[-view view_name]
[-delay_corner dc_corner_name]
[-cell]
[-net]
-power_domain power_domain_name_list
```

You can perform path-based inter power domain (IPD) timing analysis by specifying derate offset value for an entire power domain(s) and specific library cell(s) on paths crossing multiple power domains. This feature is supported for path-based analysis (PBA) mode only.

The `set_aocv_interface_path_offset` command allows you to specify the derate offset value for path segments crossing multiple power domains. The software adds the derate offset value to the AOCV derate value if the combined (AOCV + IPD offset) derate number causes pessimistic PBA timing slack value.

Parameters

`-cell`	It is a boolean option. When the -cell option is specified, intent is to apply “derate offset” only for that cell objects. In absence of these options, derate offset value will be applied to both cell and net type of objects
`-delay_corner dc_corner_name`	Specifies a list of delay corners.
derate_offset_value	Specifies value of derate offset.
lib_cell_list	Specifies a list of library cells for a specific interface.
`-net`	It is a boolean option. When the -net option is specified, intent is to apply “derate offset” only for that net objects. In absence of these options, derate offset value will be applied to both cell and net type of objects
`-power_domain power_domain_name_list`	Specifies a list of power domains.
`-view view_name`	Specifies a list of views.

Examples

- The following example specifies the offset derate value of `-0.6` for the power domain `{PD3 PD4}`:
  set_aocv_interface_path_offset -0.6 -power_domain {PD3 PD4} -delay_corner V1_PM1_dc

set_aocv_stage_weight

```
set_aocv_stage_weight
stage_weight
port_list
[-early]
[-late]
[-input]
[-output]
```

While performing advanced on-chip variation (AOCV) analysis of a block in standalone mode, the command allows you to specify the stage weight seen at the port(s) when it is modeled as part of the top level design. The applied stage weights affect the internal weight count of the block for the IO paths. The stage weights can be applied only on ports.

You can view stage weights in the `report_timing` output under the `Stage Weighting` column.

The `set_aocv_stage_weight` command can be used in multi-mode multi-corner mode. The command can be used as part of SDC file.

Parameters

`-early`	Specifies weight for early paths.
`-input`	Specifies input weight for the ports.
`-late`	Specifies weight for late paths.
`-output`	Specifies output weight for port.
`port_list`	Specifies a list or collection of input, output, and bidi ports.
`stage_weight`	Specifies value of stage weight.

Examples

- The following example specifies the stage weight value as `25.5` for port `out0`:

  ```
  set_aocv_stage_weight 25.5 out0
  ```

Related Commands

- `reset_aocv_stage_weight`
- `report_timing`

set_case_analysis

```
set_case_analysis
{0 | 1 | zero | one | rising | falling | rise | fall | non_switching}
pin_or_port_list
```

Sets constant or transitional values to a list of pins or ports for use by the timing engine. The specified constants or transitional values are valid only during timing analysis and do not alter the netlist. If there is a conflict between the specified `set_case_analysis` value and the logical value from the netlist, the value from the `set_case_analysis` is used in timing analysis.

Parameters

`0 | 1 | zero | one`	Specifies a logic value of `0` or `1`, which is propagated through combinational logic to disable or enable parts of logic.
`rising | falling | rise | fall | non_switching`	Specifies a valid transition value to be considered by the timing engine. For example, if a rising value is specified at a certain pin or port, only the rising signal transition from the pin or port is considered for timing analysis and the opposite transition type (falling) is ignored.
`pin_or_port_list`	Specifies a list of ports or pins on which to apply the constant or transitional value. Specify multiple pins by enclosing them with curly braces `{}`and separating the pin names with white space. The *`pin_or_port_list`* argument can be a collection.

Examples

- The following command sets the value of pin i4/A to `1` for use by the timing engine: set_case_analysis 1 i4/A
- The following example shows that the pins P1/P2/B and P1/P3/CI are considered only for a rising transition. The falling transition on these pins is ignored: set_case_analysis rising {P1/P2/B P1/P3/CI}

Related Information

- `check_timing -type constant_collision`
- `report_case_analysis`
- `report_timing`
- `set_disable_timing`
- `set_false_path`

set_clock_exclusivity

```
set_clock_exclusivity
[-help]
pin_name
[-exclude_opposite_polarity]
[-exclude_related_input_pins]
[-exclude_same_polarity]
[-group clock_list]
```

Controls filtering of mutually-exclusive clocks at strategic points in the design, e.g., the output of clock multiplexers.

Parameters

`-help`	Prints out the command usage.
`-exclude_opposite_polarity`	Isolates positive and negative phases of the clocks in the same group. By default, the positive and negative phases of the clocks in the same group that arrive at a strategic point will not be isolated from each other.
`-exclude_related_input_pins`	Allows comparison of clocks going through the same input pins of the specified gate.
`-exclude_same_polarity`	Allows comparison of clocks going through the same launch or capture paths.
`-group` *clock_list*	Defines a set of clocks that are exclusive to the clocks defined in all other groups.
pin_name	Specifies the name of the pin.

Examples

- The following command will cause downstream analysis from the mux to isolate interactions between the groups:
  ```
  set_clock_exclusivity -group {CLK1 CLK2} -group {CLK3 CLK4} U1/Y
  ```

  This assertion will cause downstream analysis from pin `U1/Y` to isolate interactions between the groups. The groups `{CLK1 CLK2}` and `{CLK3 CLK4}` are allowed, but all other combinations are considered logically exclusive.

Related Commands

- `create_clock`
- `reset_clock_gating_check`

set_clock_gating_check

```
set_clock_gating_check
[-help]
[object_list]
[-fall]
[-hold hold_value]
[-rise]
[-setup setup_value]
[-high | -low]
```

Specifies or overrides the default setup and hold values for clock gating checks.

Parameters

`-high | -low`	Specifies that the non-controlling value of the clock is high or low. Timing analysis performs the check on the non-controlling value of the clock. By default, the software determines the non-controlling value of the clock using information from the gate's logic. For example, for OR gates the non-controlling value is low. For complex gates, such as XOR and MUX, the clock is never controlling and it does not perform checks unless you specify either the `-high` option or the `-low` option. The specified value takes precedence over the default values. This option can only be used with pins or instances. **Note**: The clock pin value is controlling if it causes the gate output to be a constant (`0` or `1`).
`-hold hold_value`	Specifies the hold value. The default is `0.0`.
`obj_list`	Specifies a list of clocks, instances, or pins. If a clock is specified, the check applies to all the clock gating gates driven by this clock. If an instance is specified, the check applies to all input pins of the instance. If the *`obj_list`* argument is not specified, the clock gating check is applied to the current design.
`-rise | -fall`	Applies the specified setup (or hold) value to the rising (or falling) signals on the specified pins. If the clock is specified or the *`obj_list`* variable is not specified, the value applies to rising or falling, setup or hold. If neither option is specified, the default is both rising and falling.
`-setup setup_value`	Specifies the setup value. The default is `0.0`.

Note: The clock gating constraint is not propagated along the clock tree.

The setup check is performed with respect to the edge of the clock signal that changes the state of the clock pin from controlling to non-controlling. For example, for AND and NAND gates, the setup check is performed with respect to the rising edge of the clock input. For OR and NOR gates, the setup check is performed with respect to falling edge of the clock input.

The hold check is performed with respect to the edge of the clock signal that changes the state of the clock pin from non-controlling to controlling. For example, for AND and NAND gates, the hold check is performed with respect to the

falling edge of the clock input. For OR and NOR gates, the hold check is performed with respect to rising edge of the clock input.

If the gate logic in unknown, setup and hold checks are performed with respect to both the rising edge and the falling edge of the clock signal and the timing analyzer uses the worst case values.

Priority Rules for Determining Setup or Hold Values

Timing analysis uses the following priority rules to determine the setup (or hold) value to use for a particular clock gating check:

1. Use the clock gating check assertion on the data pin if it exists
2. Otherwise, use the clock gating check assertion on the clock pin if it exists
3. Otherwise, use the clock gating check assertion on the clock waveform for the clock pin if it exists
4. Otherwise, use the clock gating check assertion, if it exists
5. If no clock gating check assertion is present, use the default setup (or hold) value of `0.0`.

These rules are demonstrated in the example given below.

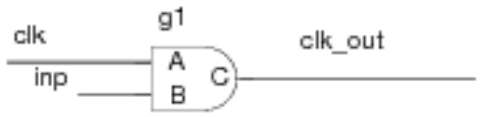
Note: This command has no effect on a clock gating check defined as a non-sequential setup or hold check in the library and reported as `ClockGatingSetup` or `ClockGatingHold`.

Examples

Here are some examples.

- The following example shows the priority rules for multiple clock gating check constraints for the gated clock shown in figure below:

  Gated Clock

  Tcl Script for set_clock_gating_check

  ```
  create_clock -name CLK1 -period 12 -waveform {0 6} clk
  ...
  # clock-gating timing check assertions:
  # On ideal clock waveform
  set_clock_gating_check -setup 0.44 [get_clocks {CLK1}]
  # On clock pin
  set_clock_gating_check -hold 0.11 [get_pins {g1/A}]
  # On data pin
  set_clock_gating_check -setup 0.66 -rise [get_pins {g1/B}]
  ```

```
set_clock_gating_check -hold 0.22 -fall [get_clocks {g1/B}]
```

Given the previous constraints, the following timing margins are used for slack calculation:

- `SETUP g1/B rise: 0.66` -- from assertion on data pin `g1/B` (rule 1)
- `SETUP g1/B fall: 0.44` -- from assertion on ideal clock `CLK1` (rule 3)
- `HOLD g1/B rise: 0.11` -- from assertion on clock pin `g1/A` (rule 2)
- `HOLD g1/B fall: 0.22` -- from assertion on data pin `g1/B` (rule 1)

Related Commands

- `create_clock`
- `reset_clock_gating_check`

set_clock_groups

```
set_clock_groups
[-name name_list]
[-comment string]
[-logically_exclusive] | [-physically_exclusive] | [[-asynchronous]
   [-allow_paths]]
[-group clock_list]
```

Defines clock groups with specified clock definitions. The software assigns all clocks that are defined after you use the set_clock_groups command to a default clock group. The software does not apply clock groupings on a master clock to the generated child clocks

If there is an overlap in the clock group specification, the software uses the following precedence to resolve the behavior:

1. Physically exclusive
2. Asynchronous
3. Logically exclusive

The software drops any explicit false path assertions between clocks that you have set as either physically or logically exclusive or asynchronous using the set_clock_groups command.

An alternative to setting the clocks as physically or logically exclusive using the `set_clock_groups` command is to use the multi-mode mechanism. Generally in an implementation flow, sequential setting of active clocks cannot be used. For concurrent analysis of different modes, you can use an multi-mode based approach to drive IPO instead of using the `set_clock_groups` command.

For more information on multi-mode multi-corner analysis, see *Importing and Exporting Designs* chapter in the *Innovus User Guide*.

Parameters

`-allow_paths`	Allows paths between asynchronous clock groups to be traced in timing analysis. These paths are blocked by default. When you use this option, the software still considers the paths to be asynchronous for signal integrity considerations. Signal integrity analysis does not support the `-allow_paths` parameter. Use the `-allow_paths` parameter with the `-asynchronous` parameter.
`-asynchronous`	Defines clock group with asynchronous clocks. Asynchronous clocks have no specified phase relationship. The software considers asynchronous nets to have infinite timing window overlap for signal integrity considerations and considers these as false paths for timing analysis. Use the `-allow_paths` parameter to trace these paths in timing analysis.
`-comment string`	Allows you to insert comments. The comments are read but are ignored with a one-time warning message. They are not used during timing analysis reporting nor are they written out during saving of constraints.

`-group` *clock_list*	Defines a set of clocks that are exclusive or asynchronous to the clocks defined in all other groups. If you define only one group, then all clocks not in the group implicitly belong to a default clock group. The software automatically adds any new clocks that you define after the `set_clock_groups` command to this default clock group.
`-logically_exclusive`	Defines clock group with clocks that do not have any functional paths between them, but coexist in the design at the same time. The software considers aggressor interactions between these clocks for signal integrity considerations, but considers these as false paths for timing analysis. **Note:** The `-logically_exclusive` parameter does not have any impact on SI analysis. For example, clocks selected via a clock mux select.
`-name` *name_list*	Specifies the names of the clock groups to be added.
`-physically_exclusive`	Defines clock group with clocks that cannot exist in the design at the same time. The software does not consider aggressor interactions between clock groups that are physically exclusive for signal integrity considerations. From the perspective of SI analysis, all attackers that are physically exclusive with the victim net are ignored for both glitch and noise-on-delay analysis. Attacker nets that belong to physically exclusive groups cannot attack together. In this case, all sets of attackers belonging to all the physically exclusive groups are evaluated and the set of nets that has the highest impact on SI analysis is chosen. For timing analysis, the software considers these clock groups as false paths. Example: multiple clocks defined on the same source pin

Examples

- The following commands assert two different clocks on the same clock root port, and uses set_clock_group to isolate them from each other:

```
create_clock -name SYSCLK -period 10 [get_ports sclk]

create_clock -name SCANCLK -period 100 -add [get_ports sclk]

set_clock_groups -name PHY_EX_SCLK -physically_exclusive

-group {SYSCLK} -group {SCANCLK}
```

- The following command declares a single bus clock to be asynchronous to all of the other on board clocks in the design (which remain in the default clock group):

```
set_clock_groups -name ASYNCH_BUS -asynchronous -group {BUSCLK}
```

- The following example shows that the clock group {clkA clkB} is asynchronous only to clock group {clkC clkD}:

```
set_clock_groups -asynchronous -group {clkA clkB} -group {clkC clkD}
```

If an extra clock clkE is defined in the design, then timing check will be allowed in between the first group to clkE

and second group to clkE.

- In the following example since only one clock group is specified with the command `set_clock_groups`command, so the specified clock group is considered to be asynchronous to all the other clocks in the design:

```
set_clock_groups -asynchronous -group {clkA clkB}

set_clock_groups -asynchronous -group {clkC clkD}
```

Related Commands

- `create_clock`
- `reset_clock_groups`

set_clock_latency

```
set_clock_latency
[-source [-early | -late]]
[-rise]
[-fall]
[-dynamic jitter_val]
[-clock clock_list]
[-min]
[-max]
[-clock_gate]
value
obj_list
```

Specifies ideal internal clock value and external clock arrival delay.

Clock value has two components:

- Source value:
  - Specified using the `-source` parameter and is the delay from the external clock generator to the clock port of the design.
  - Applied when the clock is either ideal or propagated.
  - `set_clock_latency` set on a clock port has higher precedence than `set_clock_latency` set on a clock waveform.
  - Specified on clock waveforms with both the `set_input_delay` command and the `set_output_delay` command.
- Network value (clock tree delay):
  - Default value type and is the delay from the clock port to the register.
  - Applied only when the clock is ideal. When the clock is propagated, this delay is replaced by the actual clock tree delays.
  - Specified on a clock waveform or a clock port is ignored if the clock is propagated.
  - `set_clock_latency` set on a timing pin has higher precedence than `set_clock_latency` set on a waveform.
  - If there are multiple `set_clock_latency` and `set_propagated_clock` assertions between the clock waveform, clock root and clock endpoint, the closest assertion to the clock endpoint takes precedence.
  - Specified on internal circuit pins or hierarchical ports and used in the fanout cone of the network value pin, when a clock is ideal.

Use the `set_clock_latency` command in two ways:

- To specify clock value for a clock waveform:
  - Use the following syntax for source value associated with a clock signal:
    `set_clock_latency [-source] [-early | -late] [-min ][-max] value obj_list`
  - Use the following syntax for network value associated with a clock signal:

```
set_clock_latency [-min] [-max] value obj_list
```

You cannot specify the `-late` and `-early` parameters for network value.

Specifying value on a port or pin overrides any previous value specified on the port pin or waveform. The *obj_list* argument is the list of the clock waveform names as specified by the create_clock command.

- To specify clock value on clock pins:
  - Use the following syntax for source delay on a clock port:
    ```
    set_clock_latency -source [-early | -late] [-rise] [-fall]
    insertion_delay obj_list
    ```
  - Use the following syntax for network delay on a clock pin:
    ```
    set_clock_latency [-min] [- max] [-rise] [-fall]
    insertion_delay -pin obj_list
    ```

When a clock is ideal, the last specification along a path is used. This is useful when you want to intentionally skew the clock tree and model the physical hierarchical clock tree.

Parameters

`-clock` *clock_list*	Associates clock value constraints, placed at the pin level, with a specific clock. This makes it possible to specify a different value per clock. **Note:** You cannot use the `-clock` parameter with clock waveform objects.
`-clock_gate`	Allows setting the local clock value on pin or port objects. This setting overwrites the existing value of clock gating cells and is used when performing timing checks against the specified pins or ports.
`-dynamic` *jitter_val*	Specifies the portion of source clock latency value that can have cycle-to-cycle variation. When calculating CPPR adjustment this portion is removed from CPPR adjustment for different clock edge checks (e.g., setup), but it remains as part of cppr adjustment for same clock edge checks (e.g., hold).
`-early | -late`	Specifies clock arrival time with respect to the early or the late time of the clock signal. In setup analysis, launch path is the late path and capture path is the early path. In hold analysis, launch path is the early path and the capture path is the late path. If neither parameter is specified, the default is both early and late. Use these parameters only with the `-source` parameter. You cannot use these parameters with network value. Specify the clock value at the given operating corner. **Note:** The `-min` and `-max` parameters refer to the operating corner and the `-early` and `-late` parameters refer to the clock arrival time at the source. Use the `-early` and `-late` parameters with the `-source` parameter to specify the four different clock latencies.
value	Specifies the value of the latency, in library units, as a floating point.

obj_list	Specifies a list of clock waveform names or a list of pins to associate with the clock value. The pin can be a port or an instance pin. Setting source value on a pin is ignored.
-max	Specifies the clock value for a particular operating corner. Apply the -max parameter to the maximum operating corner.
-min	Specifies the clock value for a particular operating corner. Apply the -min parameter to the minimum operating corner.
-rise/-fall	Specifies the -rise or -fall parameter to the waveform at register clock pins for ideal value and to the waveform at the source for source value. Ideal network value does not change polarity when crossing negative unate arcs.
-source	Specifies a source value. If you do not specify the -source parameter, network value is applied.

Examples

- The following set of commands enable you to create a clock using the create_clock command and apply value to the created clock using the set_clock_latency command:

```
create_clock CLK2 -period 20 [get_ports {clkA clkB clkD}]
set_clock_latency -source 2 {clkB clkD} # Define port settings
set_clock_latency -rise 5 {clkB clkD} # Define port settings
set_clock_latency -early -late -source CLK1 # Define waveform settings
set_clock_latency 3 -early -late -source CLK1 # Define waveform settings
```

Related Information

- create_clock
- set_clock_transition
- set_clock_uncertainty
- set_input_delay
- set_output_delay
- set_propagated_clock
- reset_clock_latency

set_clock_sense

```
set_clock_sense
[-positive | -negative | -stop_propagation | -logical_stop_propagation | -stop type_list]
[-clocks clock_list]
pin_or_port_list
```

Selects which phase of the clock to filter at the specified point.

Parameters

`-help`	Prints out the command usage.
`-clocks clock_list`	Specifies a list of clocks passing through a specified point to which the constraint applies. When this parameter is not specified, the filtering of clock phase will be applicable to all the clocks passing through that specified point. This parameter must be used with a specified pin name and one of the parameters : `-positive`, `-negative`, `-stop_propagation`, `-logical_stop_propagation`, or `-stop`.
`-logical_stop_propagation`	Stops the propagation of specified clocks passing through the specified pins.
`-negative`	Specifies that positive phase filtered clocks pass through pins in the *pin_list*.
pin_or_port_list	Specifies the list or collection of cell output pins. The unateness specified by the set_clock_sense command is applied to clocks propagating through the specified point.
`-positive`	Specifies that negative phase filtered clocks pass through pins in the *pin_list*.
`-stop_propagation`	Stops the propagation of specified clocks and clock source data paths passing through the specified pins.

`-stop` *`type_list`*	Applies clock sense to object types specified in the *`type_list`*. The type list options are - `all`, `clock`, `data`, `clock_source_data`, `default_data`. These are explained below: • `all`: Stops all phases from propagating. This is equivalent to specifying both the `clock` and `data` options. • `clock`: Stops clock phases launched by clock(s) specified in the *`clock_list`* from propagating, and is equivalent to existing `-logical_stop_propagation`. • `data`: Stops all data phases launched by clock(s) specified in *`clock_list`*, starting from the pins/ports specified in *`pin_or_port_list`*. The objects in the *`pin_or_port_list`* can be selected anywhere in the design - clock/data network. • `clock_source_data`: Stops only clock source data phases from propagating. The combination of `clock_source_data` and `clock` is equivalent to the existing `-stop_propagation` option. • `default_data`: Stops only default data phases (representing constrained paths) from propagation. This option cannot be specified along with the *`-clock clock_list`*. The `-stop` parameter extends the capability of the `set_clocks_sense` command to selectively block data timing phases. This parameter provides a more generalized use model for selecting among all types of paths that should be filtered. The `-stop` parameter is mutually exclusive with the `-positive`, `-negative`, `-stop_propagation`, and `-logical_stop_propagation` options.

Examples

- The following command stops the propagation of clock CLK1 through the pin CLKBUF1/Z:
  ```
  set_clock_sense -logical_stop_propagation -clocks CLK1 CLKBUF1/Z
  ```
- The following command allows the positive phase clock CLK1 to pass through CLKBUF1/Z and filters out the negative phase clock CLK1:
  ```
  set_clock_sense -positive -clocks CLK1 CLKBUF1/Z
  ```
- The following command stops clock phases starting from pin1:
  ```
  set_clock_sense pin1 -stop {clock}
  ```
  or
  ```
  set_clock_sense pin1 -logical_stop_propagation
  ```
- The following command stops data phases based only on CK1 and CK2 starting from pin1:
  ```
  set_clock_sense pin1 -stop {data} -clocks {CK1 CK2}
  ```
- The following command stops only the clock source data phases starting from pin1:
  ```
  set_clock_sense pin1 -stop {clock_source_data}
  ```
- The following command stops the clock and clock source data phases:
  ```
  set_clock_sense pin1 -stop {clock clock_source_data}
  ```
  or
  ```
  set_clock_sense pin1 -stop_propagation
  ```

Related Commands

- reset_clock_sense

set_clock_transition

```
set_clock_transition
[-rise]
[-fall]
[-min]
[-max]
[-min |-max]
transition_time
clock_list
```

Specifies the slew time of register clock pins. This is useful for pre-layout static timing analysis before clock tree synthesis (CTS) is performed since net delays can be inaccurate with large fanout nets.

Use this command only with ideal clocks. In propagated mode, the software calculates slew times on register clock pins.

Parameters

`-help`	Prints out the command usage.
`clock_list`	Specifies a list of clocks only with ideal mode in the design, which affects the slew times on all register clock pins in the transitive fanout of the specified clocks. Specified `set_clock_transition` values on register clock pins in the fanout of a clock with propagated latency are ignored.
`-rise | -fall`	Specifies the rising or falling edge of the register clock pins on which the slew time is asserted.
`-min | -max`	Specifies the clock slew time for minimum or maximum process corner slew time.
`transition_time`	Specifies the slew time (transition time) of the clock pins.

Example

- The following commands specify a `2.0` rise slew time and a `1.0` fall slew time for register clock pins clocked by `CLKA`: `set_clock_transition -rise 2.0 [get_clocks CLKA] set_clock_transition -fall 1.0 [get_clocks CLKA]`

Related Information

- report_clocks
- set_clock_latency
- set_clock_uncertainty
- set_input_delay
- set_propagated_clock

set_clock_uncertainty

```
set_clock_uncertainty
uncertainty_value
[-setup]
[-hold]
[-half_cycle_jitter | -full_cycle_jitter]
[-rise | -fall]
{  {-from | -rise_from | -fall_from} clksig_from_list
   {-to | -rise_to | -fall_to} clksig_to_list
 | clock_or_clock_pin
}
```

Specifies the clock uncertainty (skew) on the clock network.

If you set multiple constraints with the `set_clock_uncertainty` command, the constraints set with the `-from* clksig_from_list`/`-to* clksig_from_list` parameters take precedence over simple constraints set on clock objects.

Clock uncertainty set on pins takes highest precedence and overrides clock based uncertainty.

The `set_clock_uncertainty` command can also support clock uncertainty for period, pulse width, skew, and separation checks. To enable this feature, you need to set `timing_enable_uncertainty_for_clock_checks` attribute to `true`. By default, the attribute is set to `false`.

Parameters

uncertainty_value	Specifies the uncertainty value.
-from | -rise_from | -fall_from *clksig_from_list*	Applies the uncertainty value to the specified list of launching clocks. **Note:** You must specify one of these parameters with one of the -to, -rise_to, or -fall_to parameters. You cannot use these parameters when *clock_or_clock_pin* is specified.
	-fall_from: Applies the uncertainty value only to the falling edge of the launching clocks.
	-from: Applies the uncertainty value to both the rising and falling edges of the launching clocks.
	-rise_from: Applies the uncertainty value only to the rising edge of the launching clocks.
-half_cycle_jitter *value* | -full_cycle_jitter *value*	Specifies the clock source jitter for half-cycle and for full-cycle paths. These parameters can accept different values, that can be specified in the same constraint. **Note**: These parameters can be used on clocks only. You cannot use them when any of the -from and -to parameter pairs are specified.

clock_or_clock_pin	Specifies a list of clocks or pins for target uncertainty. The uncertainty value is applied to the registers clocked by the target waveform, or register pins in the fanout of the specified pin. You can specify a list of clock names, or clock root pin or port names relative to the current module. You also can specify a collection of clocks, or clock root pins or ports. **Note:** You cannot use this parameter when any of the `-from` and `-to` parameter pairs are specified.
`-rise | -fall`	Specifies the rising or falling edge at the capture clock pin on which the uncertainty is applied. *Default*: Applies uncertainty on both edges **Note:** You must use these parameters with the `-to` and `-from` parameters.
`-setup/-hold`	Specifies whether the clock uncertainty applies to setup or hold checks. *Default*: Applies to both setup and hold check
`-to | -rise_to | -fall_to` *clksig_to_list*	Applies the uncertainty value to the specified list of capture clocks. **Note:** You must specify one of these parameters with one of the `-from`, `-rise_from`, or `-fall_from` parameters. You cannot use these parameters if *clock_or_clock_pin* is specified.
	`-fall_to`: Applies the uncertainty value only to the falling edge of the capture clocks.
	`-rise_to`: Applies the uncertainty value only to the rising edge of the capture clocks.
	`-to`: Applies the uncertainty value to both the rising and falling edges of the capture clocks.

Examples

- The following command gives all paths starting from registers clocked by `clk1` to registers clocked by the rising edge of `clk2` a hold uncertainty of `2.0`:

  ```
  set_clock_uncertainty 2.0 -from clk1 -to clk2 -rise -hold
  ```

  In case of multiple constraints, the actual uncertainty applied by the software depends on which constraint matches the specific path in question.

- The following command sets an uncertainty on the `clk_in` clock root pin that is applied to all the register pins in the fanout of `clk_in`:

  ```
  set_clock_uncertainty 1.2 clk_in
  ```

- The following command specifies an inter-clock uncertainty of `0.2` between clock waveforms `clk1` and `clk2`:

  ```
  set_clock_uncertainty 0.2 -from clk1 -to clk2
  ```

- The following command specifies a target uncertainty of `0.5` for all registers clocked by the clock waveform

```
ideal_clk1:
set_clock_uncertainty 0.5 ideal_clk1
```

Related Information

- report_clocks
- set_clock_latency
- set_clock_transition
- reset_clock_uncertainty

set_data_check

```
set_data_check
[-help]
<check_value>
[-clock <clock_object>]
[-hold]
[-setup]
{-from <pin_or_port_list> | -rise_from <pin_or_port_list> | -fall_from <pin_or_port_list>}
{-to <pin_or_port_list> | -rise_to <pin_or_port_list> | -fall_to <pin_or_port_list>}
```

Sets data-to-data checks using the specified values of setup and hold time. Data-to data checks are non-sequential checks that are designed to constrain a data signal with respect to another data signal.

In a normal sequential check, a data signal is constrained with respect to a clock signal. In these cases single-cycle timing is used - allowing one full clock cycle for the data transfer. In a non-sequential data-to-data check, there is no clock reference. Therefore zero-cycle timing rules are used. You can use the set_multicycle_path constraint to adjust the cycle accounting rules.

Parameters

`-clock` *clock_object*	Specifies the name of the clock that triggers the data event for the related (`-from`) pin. In the case where there are multiple data events triggered by different clock phases arriving at the `-from` pin, the `-clock` parameter allows you to specify which data event should be used for the "related pin" side of the timing check.
{`-from` <*pin_or_port_list*> | `-rise_from` <*pin_or_port_list*> | `-fall_from` <*pin_or_port_list*>}	Specifies the pins or ports that are set as related pins for data-to-data checks. The *pin_or_port_list* argument can contain either object IDs or hierarchical names relative to the current module. Use only one `-from`, `-rise_from`, or `-fall_from` parameter per command. By default, the `-from` parameter checks both the rising and the falling edges. Using `-rise_from` parameter checks at the rising edge of the signal on the from pins. Using `-fall_from` parameter checks at the falling edge of the signal on the from pins. **Note:** You cannot use the `-from` parameter to specify multiple objects.
`-setup` | `-hold`	Specifies whether the data check value is for setup or hold check. If you do not specify either `-setup` or `-hold` option, the value applies to both setup and hold.

{-to <*pin_or_port_list*> | -rise_to <*pin_or_port_list*> | -fall_to <*pin_or_port_list*>}	Specifies the pins or ports that are set as constrained pins for data-to-data checks. The *pin_or_port_list* argument can contain either object IDs or hierarchical names relative to the current module. Use only one -to, -rise_to, or -fall_to parameter per command. By default, the -to parameter constrains both the rising and the falling edges. Using -rise_to parameter constrains the rising edge of the signal on the to pins. Using -fall_to parameter constrains the falling edge of the signal on the to pins.
check_value	Specifies the value of the setup or hold time for the check.

Examples

- The following example demonstrates how the set_data_check command can be used to constrain signals: Consider two control signals, say XE and SE, going to a macro block. Both of these are derived from the same state machine, and hence launched from the same clock domain. You want XE to reach the macro earlier than SE. The set_data_check constraint can meet this design intent, by applying data checks on non-sequential signals, without setting path specific limits using the following command:

  ```
  set_data_check -from [get_pins {sub/XE}] -to [get_pins {sub/SE}] -setup 0.5
  ```

  The set_data_check command establishes data-to-data checks using similar ideas of setup and hold time as in sequential checks, with the -from parameter denoting the reference signal (the clock in a traditional setup/hold timing check), and the -to parameter indicating the signal being constrained.

Related Commands

- reset_data_check

set_disable_clock_gating_check

```
set_disable_clock_gating_check
pin_or_inst_list
```

Disables clock gating checks for the specified instances or pins.

Parameters

pin_or_inst_list	Specifies the list of pins or instances that are to be disabled.

Examples

- The following command disables clock gating checks for the specified instance:
  ```
  set_disable_clock_gating_check [get_cells clockgate1]
  ```

Related Commands

- reset_disable_clock_gating_check

set_disable_timing

```
set_disable_timing
[-from pin_name -to pin_name]
obj_list
```

Disables timing propagation through the specified arcs, or a collection of instance-specific arcs that are created using the `get_arcs` command.

Parameters

`-from pin_name -to pin_name`	Specifies the arc that is to be disabled. Specify the `-from` and `-to` options together. You can specify the `-from` and `-to` parameters when the *`obj_list`* contains instance or library cell objects. Do not specify the `-from` and `-to` parameters when the *`obj_list`* includes instance pin or library cell pin objects. The `-from` and `-to` options are optional. If not specified, all cell arcs found in the instance or library cell objects are disabled.
`obj_list`	Specifies the instance, instance pin, library cell, library cell pin objects, collection of library arcs, or collection of timing arcs. If the *`obj_list`* includes instance pin or library cell pin objects, all cell arcs attached to the specified pin objects are disabled. If the *`obj_list`* contains instance or library cell objects, and the `-from` and `-to` parameters are not specified, all arcs found in the specified instance or library cell objects are disabled.

Examples

- The following command disables the arcs that lead to the output pins of the `i1`, `i2`, and `i3` instances. Note that timing checks at the inputs of `i1`, `i2`, and `i3` are still active. `set_disable_timing [get_cells {i1 i2 i3}]`
- The following command disables the arc from pin `B` to pin `Z` in the `i1` and `i2` instances, and the arc from pin `A` to pin `Z` in the `i3` instance: `set_disable_timing -from B -to Z [get_cells {i1 i2}]`
  `set_disable_timing -from A -to Z [get_cells {i3}]`
- The following example disables timing check arcs from pin `CLK` to pin `D` of the `ff1` instance: `set_disable_timing -from CLK -to D [get_cells ff1]`
- The following example disables the outgoing cell arcs from instance input pin `i1/B`: `set_disable_timing [get_pins i1/B]`
- The following example disables all arcs found in any instances whose reference cell is `AN2`: `set_disable_timing [get_lib_cells AN2]`

- The following example disables a collection of timing arcs:

```
set arcs [get_lib_arcs -from typical/MX2X1/S0 -to typical/MX2X1/Y]
set_disable_timing [index_collection $arcs 3]
```

- The following example disables arcs created using `filter_collection` on library timing arcs:

```
set arc [filter_collection $arcs {when == "((A)&(!(B)))"}]
set_disable_timing $arc
```

Related commands

- `report_timing`
- `reset_disable_timing`

set_dont_touch

```
set_dont_touch
obj_list
[true | false]
```

Prevents the specified object from being modified during optimization. However, the specified object can be moved during optimization.

When you specify a net, it cannot be buffered or become a part of the logic cone restructuring. However, you can resize the instances connected to this net because the `dont_touch` attribute set of the net does not propagate on the connected instances.

Parameters

`true | false`	Sets or removes the `set_dont_touch` attribute for the specified *`obj_list`*. *Default:* `true`
`obj_list`	Specifies the list of instances, nets, library cells, or submodules.

Examples

- The following command prevents the specified instance from being modified:
  ```
  set_dont_touch [get_cells {TWA/FF1}]
  ```
- The following commands can be used to prevent any ECO inside RESULTS_CONV_INST hierarchical instance:
  ```
  set_dont_touch [get_cells RESULTS_CONV_INST/*] true
  set_dont_touch [get_nets RESULTS_CONV_INST/*] true
  ```

  The `set_dont_touch RESULTS_CONV_INST/* true` command will return `[get_cells RESULTS_CONV_INST/*]` only. This is because the software looks into the first match (type of object). If a match is not found, then the successive objects are searched.

Related Commands

- `get_cells`

set_dont_touch_network

```
set_dont_touch_network
[-help]
obj_list
[-no_propagate]
```

Prevents the combinational path connected to the given clocks, pins, or ports from being modified.

Parameters

`-help`	Prints out the command usage.
`object_list`	Specifies the objects - clocks, pins, or ports - to be preserved.
`-no_propagate`	Preserves all the nets driven by the specified pin up to the first leaf-cell input (combinational or sequential). If this parameter is omitted, then preserving stops at the first sequential cell in the network.

Examples

- The following command prevents the combinational path connected to the specified pin from being modified:
  ```
  set_dont_touch_network [get_pins and1/z]
  ```

Related Commands

- `get_pins`

set_dont_use

```
set_dont_use
[-help]
lib_cell_list
[true | false]
```

Prevents the specified design, hierarchical module, or library cells from being used during optimization. However, optimization is allowed to size out don't use cells if instances are already there in the initial netlist.

Parameters

lib_cell_list	Specifies the list of library cells.
true | false	Sets or removes the set_dont_use attribute for the specified object list. Default: true

Examples

- The following command prevents the specified library cell from being used during optimization:
  ```
  set_dont_use [get_lib_cells {and2}]
  ```

Related Commands

- set_clock_sense

set_drive

```
set_drive
[-rise] [-fall]
[-max] [-min]
resistance_value
port_list
```

Sets the drive resistance on the specified input and/or bidirectional ports of the top cell.

The drive resistance models the cell driving the port. This resistance is used to compute the load dependent delay and the slew at the specified ports. The load-dependent delay is added to the arrival times.

The load-dependent delay and the slew are computed by multiplying the drive resistance by the wire capacitance of the net connected to the port.

Use the `report_ports` command to view drive resistances asserted on ports.

Parameters

`-fall`	Specifies that the drive resistance is to be used for the falling transition at the specified ports. By default, if neither the `-rise` nor the `-fall` option are specified, the drive resistance applies to both the rising transition and the falling transition.
`-min`	Specifies that the drive resistance is to be used to compute the early path delay. By default, if neither the `-min` option nor the `-max` option is specified, the drive resistance applies to both the early path delay and the late path delay.
`-max`	Specifies that the drive resistance is to be used to compute the late path delay. By default, if neither the `-min` option nor the `-max` option is specified, the drive resistance applies to both the early path delay and the late path delay.
port_list	Specifies a list of input or bidirectional ports on which the drive resistance is set.
resistance_value	Specifies the value of the drive resistance. This value must be greater than or equal to zero. The drive resistance is in units found in the target technology library.
`-rise`	Specifies that the drive resistance is to be used for the rising transitions at the specified ports. By default, if neither the `-rise` option nor the `-fall` option are specified, the drive resistance applies to both the rising transition and the falling transition.

Example

- The following command sets a drive resistance of 2.0 for the early path, the late path, the rising and the falling transitions at the in1 and in2 ports and shows how drive resistances are reported using the report_ports command as shown in the example below:

```
> set_drive 2.0 [get_ports {in1 in2}]
> report_ports -type drive_resistance -pin [get_ports {in1 in2}]
```

  The following report is displayed:

```
-------------------------------------------------------------------
Early Late
-------------------------------------------------------------------
Pin     Dir   Assertion       Clock   Rise   Fall   Rise   Fall
Name                          Name
-------------------------------------------------------------------
in1    IN    drive_resistance *(D)(P) 2.000   2.000  2.000  2.000
in2    IN    drive_resistance *(D)(P) 2.000   2.000  2.000  2.000
-------------------------------------------------------------------
```

Related Commands

- all_inputs
- report_ports
- set_driving_cell
- set_input_transition
- set_load

set_driving_cell

```
set_driving_cell
[-library library_name]
-lib_cell cell_name
[-pin pin_name]
[-from_pin from_pin_name]
[-rise]
[-fall]
[-min]
[-max]
[-multiply_by factor]
[-no_design_rule]
[-input_transition_rise rise_slew_value]
[-input_transition_fall fall_slew_value]
port_list
```

Models the drive capability of an external driver connected to the input port. Use this command only for timing analysis. The command does not affect the electrical properties of the design, which means that the capacitance of the output pin of the driving cell does not add to the total capacitance of the input port.

To model the drive capability of the external driver, the timing analyzer computes an offset to the arrival time of an input, and changes the slew time used to compute the delay for the next cell. The timing analyzer computes the offset by subtracting the zero load delay from the total delay of the cell. The `set_driving_cell` command uses the offset value to identify a cell from the library to drive the specified input port(s).

This command overrides the `set_drive` command per port or per net, if it is the last command applied. Also, if the `set_driving_cell` command is set for a port, the `set_input_transition` command is ignored for that port. If you do not define drive using the `set_driving_cell` command, the software uses the default value defined in the configuration file. The default value is `0.1ps`.

Parameters

`-fall`	Annotates only the falling drive assertion. The default is to annotate both rise and fall delays.
`-from_pin` *from_pin_name*	Specifies the name of the input port of the driver cell that has an arc to the output port of the driver cell, which is connected to the input port of the design. This option is required only if there are multiple timing arcs in the driver cell from its input ports to its output ports, and the arcs from those pins have different drive characteristics. If you do not specify the `-from_pin` option, the default is to use the first timing arc found on the library cell.
`-input_transition_fall` *fall_slew_value*	Specifies the slew value for the falling signal at the input of the drive cell.

`-input_transition_rise` *rise_slew_value*	Specifies the slew value for the rising signal at the input of the drive cell.
`-lib_cell` *cell_name*	Specifies the name of the driver cell. This parameter supports wildcard patterns and collections.
`-library` *library_name*	Specifies the name of the library that contains the driving cell.
`-max`	Annotates only the late drive assertion. The default is to annotate both early and late.
`-min`	Annotates only the early drive assertion. The default is to annotate both early and late.
`-multiply_by` *factor*	Specifies a multiplier for the drive.
`-no_design_rule`	Does not allow transfer of design rules from the driving cell to the port. If you do not specify this option, the design rules (such as `max_capacitance`) of the library pin are applied to the port.
`-pin` *pin_name*	Specifies the name of the output port of the driver cell that drives the signal at the input port of the design. This option is required if there are multiple output ports on the driver. If there is only one output port on the driver cell, then using this option is not necessary.
port_list	Specifies the list of ports driven by the driver cell.
`-rise`	Specifies only the rising drive assertion is annotated.The default is to annotate both rise and fall delays.

Examples

The following command sets 'AND' cell (arc B -> Y) as the driving cell for the input port IN1:

```
> set_driving_cell -lib_cell AND -from_pin B -pin Y IN1
```

Related Information

- `all_inputs`
- `report_ports`
- `set_drive`
- `set_input_transition`

set_false_path

```
eset_false_path
[-hold | -setup]
[-rise]
[-fall]
{[{-from | -rise_from | -fall_from} from_list]
[{-through | -rise_through | -fall_through} through_list]
[{-to | -rise_to | -fall_to} to_list]}
```

Identifies false paths in a design, and breaks or disables specific instance timing arcs in a design.

To break combinational loops, use the set_disable_timing command instead of the `set_false_path` command.

The `set_false_path` command differs from the `set_disable_timing` command as follows:

- `set_false_path` command selectively disables the arrival time information depending on the precedence rules.
- `set_false_path` command does not affect constant values.
- `set_disable_timing` command physically snips a timing arc such that neither arrival times nor constant values propagate through it.
- `set_disable_timing` command is applied in all cases to valid timing arcs.

The `set_false_path -from fpin` command disables all timing paths whose source pin is an `fpin`. Similarly, the `set_false_path -to tpin` command disables all timing paths whose sink pin is a `tpin`. Using the `set_false_path -from fpin -to tpin` command disables all timing paths whose source is `fpin` and sink is `tpin`.

Specify a group of pins by enclosing the group with curly braces (`{}`) within one `set_false_path` command.

To set a false path for any path beginning at `p1` or `p2` and terminating at `t1` or `t2`, group the pins as follows:

```
set_false_path -from {p1 p2} -to {t1 t2}
```

Note: This method of pin grouping reduces the number of exceptions. Having large numbers of exceptions adversely impacts the run time of the timing analysis commands.

Parameters

`{-from | -rise_from | -fall_from} from_list`	Specifies a list of timing paths start points. The from_list argument can contain either object IDs or hierarchical names relative to the current module. The pins can be on intermediate hierarchical boundaries. The from_list argument can be a collection. The valid start points are: • Input port • Bidirectional port • Clock pin of sequential cell Clock pin of sequential cell is one of the following: • Structural clock pin that is part of latch or flip-flop. • Inferred clock pin in cells like macro cells. A pin is an inferred clock pin if it is associated with one of the following: • Reference end of setup check • Reference end of hold check • Reference end of removal check • Reference end of recovery check • Minimum period check • Minimum pulse width check • Pin associated with set_input_delay Specify one of the following:
	`-from`: Applies the false path constraint to all paths originating from the from_list list. If a clock is specified as the -from object, all primary inputs and registers clocked by that clock are used as start points. By default, the -from option applies the constraint to both the rising and the falling edges.
	`-rise_from`: Applies the false path constraint at the rising edge of the signal on the from pins.
	`-fall_from`: Applies the false path constraint at the falling edge of the signal on the from pins.
`-hold/-setup`	Sets false paths for either setup or hold analysis. Note: If you do not specify -setup or -hold, the software sets false paths for both setup and hold analysis.
`-rise | -fall`	Applies the assertion at the rising edge of the signal on the path endpoints. Applies the assertion at the falling edge of the signal on the path endpoints.

`{-through | -rise_through | -fall_through} through_list`	Specifies a list of timing paths through points. The through_list argument can contain either object IDs or hierarchical names relative to the current module. The pins can be on intermediate hierarchical boundaries. The through_list argument can be a collection. Specify one of the following:
	`-through`: When you use this parameter without using the -from or -to options, the false path constraint is set for all paths that go through the specified pins. When you use it with both the -from and -to options, any path starting at any pin on the -from list and going through any point on the -through pin list and going to any point on the -to list is affected by the constraint. By default, the -through option applies the constraint to both the rising and the falling edges.
	`-rise_through`: Applies the assertion at the rising edge of the signal on the through pins.
	`-fall_through`: Applies the assertion at the falling edge of the signal on the through pins.
`{-to| -rise_to | -fall_to} to_list`	Specifies a list of timing paths end points. The to_list argument can contain either object IDs or hierarchical names relative to the current module. The pins can be on intermediate hierarchical boundaries. The to_list argument can be a collection. The valid end points are: • Output port • Bidirectional port • Data pin of sequential cell Data pin of sequential cell is one of the following: • Structural data pin that is part of latch or flip-flop. • Inferred data pin in cells like macro cells. A pin is an inferred data pin if it is associated with one of the following: • Signal end of setup check • Signal end of hold check • Signal end of non-sequential setup check • Signal end of non-sequential hold check • Signal end of removal check • Signal end of recovery check • Pin associated with set_external_delay • Data pin of clock gating cell Select from one of the following:

	`-to`: Applies the false path constraint to all paths ending in the -to pin list. If a clock is specified as the -to object, all primary outputs and registers clocked by that clock are used as endpoint. By default, the -to option applies the constraint to both the rising and the falling edges.
	`-rise_to`: Applies the false path constraint at the rising edge of the signal on the to pins.
	`-fall_to`: Applies the false path constraint at the falling edge of the signal on the to pins.

Examples

- The following command disables hold checking from `ff1` to `ff2`:

  ```
  set_false_path -hold -from ff1 -to ff2
  ```

- The following command sets all timing paths from `ff1/CK` to `D` input of `ff2` that pass through `u1/Y` and one or more of `u2/Y` and `u3/Y` as false paths:

  ```
  set_false_path -from ff1/CK -through {u1/Y} -through {u2/Y u3/Y} -to ff2/D
  ```

- The following command sets all timing paths from the rise of `ff1/CK` to `ff2/D` through one or more of `u1/Y` and `u4/Y` as false paths:

  ```
  set_false_path -rise_from ff1/CK -through {u1/Y u4/Y} -to ff2/D
  ```

Related Information

- `set_disable_timing`
- `set_max_delay`
- `set_min_delay`
- `set_multicycle_path`
- `reset_path_exception`

set_fanout_load

```
set_fanout_load
[-help]
value
port_list
```

Specifies the fanout load on a set of ports.

The library instance pins can have fanout loads defined in the technology library. The fanout does not have specific units so it is assumed that a fanout as used by the constraints is consistent with the fanout defined in the technology libraries. If a pin or port has no fanout attribute, either from the constraints or from the technology library, then the fanout load will be zero. The fanout load is used only for design rule checking and does not impact the timing results.

Parameters

`-help`	Prints out the command usage.
value	Specifies the fanout load value for the port or ports.
port_list	Specifies the list of ports for which the fanout load is applied.

Examples

- The following command sets the fanout load of port `out` to `2`:

  ```
  set_fanout_load 2 out
  ```

- The following command sets the fanout load of ports `in1` and `in2` to `1`:

  ```
  set_fanout_load 1 {in1 in2}
  ```

- The following command sets the fanout load of element `3` of port `out` to `12`:

  ```
  set_fanout_load 12 { out[3] }
  ```

- The following command sets the fanout load of all the ports to `2`:

  ```
  set_fanout_load 2 [get_ports *]
  ```

Related Information

- `report_net`
- `report_ports`
- `set_max_fanout`
- `report_constraint`

set_ideal_latency

```
set_ideal_latency
[-help]
[-min]
[-max]
[-rise]
[-fall]
value
pin_or_port_list
```

Specifies the ideal latency to use for input or output leaf-cell pins and top-level ports that are part of an ideal network. These set a point delay on the specified object.

Note: You should specify `set_ideal_latency` with the `set_ideal_network` command to view the ideal latency result.

Parameters

`-help`	Prints out the command usage.
`-fall`	Specifies the ideal latency for the falling edge.
`value`	Specifies the ideal latency value.
`-max`	Specifies the ideal latency for the maximum corner.
`-min`	Specifies the ideal latency for the minimum corner.
`pin_or_port_list`	Specifies a list of objects - instance pins or ports.
`-rise`	Specifies the ideal latency for the rising edge.

Examples

- The following command adds an ideal latency of 1.5ns to the input pin A of cell FE_OFC413_scan_se:
  `set_ideal_latency 1.5 FE_OFC413_scan_se/A`

  Please note that the above command will only be valid if the pin is part of an ideal network.

  Now, this ideal latency value can be observed in the output of the following `report_timing` command:
  `report_timing -though FE_OFC413_scan_se/A`

```
Path 1: MET Setup Check with Pin RESULTS_CONV_INST/r697_reg[14]/CK
Endpoint: RESULTS_CONV_INST/r697_reg[14]/SE (v) checked with trailing edge of 'm_rcc_clk'
Beginpoint: scan_se (v) triggered by leading edge of 'm_ram_clk'
Path Groups: {m_rcc_clk}
Analysis View: scan_fast_RCMIN
Other End Arrival Time          20.679
- Setup                         1.123
+ Phase Shift                   0.000
+ Cycle Adjustment             40.000
- Uncertainty                   0.400
= Required Time                59.156
- Arrival Time                  8.000
= Slack Time                   51.156
Clock Rise Edge                       0.000
+ Input Delay                        4.000
+ Ideal Latency                      1.500
= Beginpoint Arrival Time            5.500
-------------------------------------------------------------
Pin                      Cell   Load    Slew   Delay Arc
                                                  Annotation
-------------------------------------------------------------
scan_se                         0.000   0.000          Ideal
FE_OFC413_scan_se/Z  BUF_X32    0.000   0.000  0.000   Ideal
FE_OFC247_scan_se/A  BUF_X16    0.000   0.000  0.000   Ideal
FE_OFC247_scan_se/Z  BUF_X16    0.000   0.000  0.000   Ideal
FE_OFC248_scan_se/A  BUF_X16    0.000   0.000  0.000   Ideal
FE_OFC248_scan_se/Z  BUF_X16    0.000   0.000  0.000   Ideal
FE_OFC249_scan_se/A  BUF_X8     0.000   0.000  0.000   Ideal
-------------------------------------------------------------
```

Related Commands

- reset_ideal_latency
- report_timing
- set_ideal_network

set_ideal_network

```
set_ideal_network
obj_list
[-no_propagate]
```

Identifies driver pins or ports as sources of an ideal network. By default, all pins and nets in the transitive fanout of the specified port or driver pin are marked as ideal. Ideal network marking normally propagates through combinational cells and stops at sequential cells.

This command is not view/mode-based - it applies to all the active views.

The `set_ideal_network` command sets 0 slew for bi-directional pins. This ensures that unnecessary DRV violations do not occur.

Parameters

`-no_propagate`	Prevents propagation through transitive fanout.
`obj_list`	Specifies a list of objects - instance pins, ports, or nets.

Related Commands

- `reset_ideal_network`

set_ideal_transition

```
set_ideal_transition
[-min]
[-max]
[-rise]
[-fall]
value
pin_or_port_list
```

Specifies the ideal transition to use for input or output leaf-cell pins and top-level ports that are part of an ideal network. This sets a point delay on the specified object.

Note: You should specify `set_ideal_transition` with `set_ideal_network` to view ideal transition result.

Parameters

`-fall`	Specifies the ideal transition for the falling edge.
`-max`	Specifies the ideal transition for the maximum corner.
`-min`	Specifies the ideal transition for the minimum corner.
`pin_or_port_list`	Specifies a list of objects - instance pins or ports.
`-rise`	Specifies the ideal transition for the rising edge.
`value`	Specifies the ideal transition time.

Related Commands

- `reset_ideal_transition`

set_input_delay

```
set_input_delay
[-clock clock_name]
[-clock_fall]
[-rise] [-fall]
[-max] [-min]
[-add_delay]
[-network_latency_included]
[-source_latency_included]
[-reference_pin pin_name]
[-level_sensitive]
value
pin_or_port_list
```

Defines the arrival time relative to a clock edge on input ports or internal input pins. This input path delay models the delay from an external register to an input port of the module.

If the input delay is not specified on an input port, it is assumed to be zero.

Parameters

`-add_delay`	Adds delay information to the existing input delay. The additional delay information can be for multiple paths relative to different clocks or clock edges leading to an input port or internal input pins. If you do not use the `-add_delay` option, the software uses the last `set_input_delay` related to a pin when there are multiple `set_input_delay` statements for the same pin.
`-clock` *clock_name*	Specifies that the delay is relative to the rising edge of the given clock. If you do not specify the `-clock` parameter, the delay is relative to time zero. To specify a delay relative to the falling edge of the clock, use the -clock_fall option with the -clock option. When you specify multiple clocks, an error occurs and the constraint is ignored.
`-clock_fall`	Specifies that the delay is relative to the falling edge of the clock. You must specify the -clock parameter with this parameter. *Default:* Delay is relative to the rising edge. You cannot specify the `-clock_fall` parameter without the `-clock` parameter.
value	Specifies the value for the path delay.
`-fall`	Specifies that the input delay refers to the falling edge of the signal at specified ports or pins. If you do not specify the `-fall` option, the input delay applies to both the edges.

`-level_sensitive`	Models input delay that is launched by a positive level sensitive latch. Use the `-clock` parameter to model a positive level sensitive latch, or the `-clock_fall` parameter to model a negative level sensitive latch. You cannot specify the `-level_sensitive` parameter without the `-clock` parameter.
`-max`	Specifies that the delay refers to the setup analysis. If you do not specify the `-max` parameter, the input delay applies to both setup and hold analysis. **Note**: If for a specific clock the maximum delay is less than the minimum delay, then during analysis the software makes the maximum delay equal to the minimum delay.
`-min`	Specifies that the delay refers to the hold analysis. If you do not specify the `-min` parameter, the input delay applies to both setup and hold analysis.
`-network_latency_included`	Specifies that the clock network latency should not be added to the input delay value. If you do not specify this option, the clock network latency of the related clock is added to the input delay value. This parameter has no effect if the specified clock is in propagated mode.
pin_or_port_list	Specifies a list of input port or pin names. The pin_or_port_list argument can be a collection.
`-reference_pin` *pin_name*	Adds source and network latency of the specified reference pin to the input delay value in propagated mode. A reference pin is a leaf pin in the fanout cone of the clock source of the clock that you specify using the `-clock` parameter. You cannot use the `-reference_pin` parameter to specify multiple pins. If specified, the command issues a warning and accepts the first pin. You cannot use the `-reference_pin` parameter with the `-network_latency_included` or `-source_latency_included` parameters. You cannot use the `-reference_pin` parameter to specify hierarchical pins.
`-rise`	Specifies that input delay refers to the rising edge of the signal at the specified ports or pins. If you do not specify the `-rise` option, the input delay applies to the both edges.
`-source_latency_included`	Specifies that the clock source latency should not be added to the input delay value. With different delay values for `-rise/-fall` and `-min/-max`, only the last specified combination of `-network_latency_included` and `-source_latency_included` parameters is honored. The specified combination can be one of the following: a) Either `-network_latency_included` or `-source_latency_included` b) Both of these c) None of these

Examples

- The following command sets an input delay of 1.0 for all inputs in the current design relative to the rising edge of the `CLK1` clock:

  ```
  set_input_delay -clock CLK1 1.0 [all_inputs]
  ```

- The following command sets an input delay of 2.0 for the input `IN1` port relative to the falling edge of the `CLK1` clock: `set_input_delay -clock CLK1 2.0 -clock_fall { IN1 }`
- The following example shows an input `IN1` port that is constrained by both the rising edge and the falling edge of the same `CLK1` clock. The `-add_delay` option prevents the second `set_input_delay` statement from overwriting the first `set_input_delay`:

  ```
  set_input_delay -clock CLK1 1.0 { IN1 }
  set_input_delay -clock CLK1 -clock_fall -add_delay 2.0 { IN1 }
  ```

- When multiple `-min` and `-max` values are specified relative to the same clock using the `-add_delay` option at the same port or pin, the worst case value for each case is used. For the `-min` case the smallest value is retained, and for the `-max` case the largest value is retained. For example, the input delay value of 1.0 is stored for the shortest path and the value of 4.0 is stored for the longest path in the following example:

  ```
  set_input_delay -min -clock CLK1 1.0 { IN1 }

  set_input_delay -min -clock CLK1 2.0 -add_delay { IN1 }

  set_input_delay -max -clock CLK1 3.0 -add_delay { IN1 }

  set_input_delay -max -clock CLK1 4.0 -add_delay { IN1 }
  ```

Related Information

- `all_inputs`
- `create_clock`
- `report_ports`
- `set_load`
- `set_output_delay`

set_input_transition

```
set_input_transition
[-help]
[-min]
[-max]
[-rise]
[-fall]
value
port_list
```

Specifies the slew time for a port on the top level module. This command is useful only in specifying a hard ramp signal as if the port were driven by a very powerful driver. The command does not change the arrival time of an input, but changes the slew time used to compute the delay of the cell on the sink of a net. If you do not define transition using the `set_input_transition` command, the software uses the default input port transition defined in the configuration file. The default value is `0.1ps`. You can also set the `delaycal_input_transition_delay attribute` to override the default value.

The `set_input_transition` command specification for clock signals does not impact setup/hold timing calculations.

Note: The `set_input_transition` command does not support internal pins. For specifying transition on internal pins use the `set_annotated_transition` command.

Note: The most recent command settings (of `set_driving_cell`, `set_input_transition`, or `set_drive`) that are applied to a port will be considered - the software will override the previous commands.

Parameters

`-help`	Prints out the command usage.
`-fall | -rise`	Specifies that the slew time should be applied to the rising edge or the falling edge of the signal (data or clock). If neither time is specified, the default is to use both.
`-min | -max`	Indicates whether the slew times are with respect to the early (hold) or the late (setup) time checks of the data signal. If neither option is specified, the default is to use both.
`port_list`	Specifies the list of ports for which the slew time is specified. **Note:** You cannot use the `set_input_transition` command to specify transition on output ports. **Note:** When you use both `set_input_transition` and `set_annotated_transition` commands for specifying transition on input ports, the worst value is used as the transition value.
`value`	Specifies the transition time.

Examples

- The following command specifies the slew time of `0.5` on port `input3`.

```
set_input_transition 0.5 input3
```

Related Commands

- `report_ports`
- `set_driving_cell`
- `set_drive`
- `set_clock_transition`

set_load

```
set_load
[-max] [-min]
[-pin_load]
[-wire_load]
[-subtract_pin_load]
capacitance_value
port_or_net_list
```

Sets the specified capacitance on the defined ports (or nets) of the top cell.

The specified capacitance overrides the capacitance from the wire load models and SPEF.

You can use the `report_ports` command to view capacitance of asserted ports. The pin load appears in the report as `port_cap` in the assertion column and the wire load appears as `port_wire_cap` in the assertion column.

In the timing report (generated using the `report_timing` command), the load value for ports is displayed as a sum of the `set_load` value on a port and the existing wire capacitance obtained from the SPEF file, and pin capacitance obtained from the library files.

Parameters

capacitance_value	Specifies the value for the capacitance. The capacitance is in units from the target technology library.
-min	Specifies the minimum capacitance. The minimum capacitance is used to compute best-case delays in BcWc analysis mode and early path delays in OCV analysis mode.
-max	Specifies the maximum capacitance. The maximum capacitance is used to compute worst-case delays in BcWc analysis mode and late path delays in OCV mode. It is also used to check the maximum capacitance design rule.
port_or_net_list	Specifies a list of ports or nets in the top cell.
-pin_load	Specifies the external pin capacitance for a port. You can use this parameter for ports only. For ports, the asserted capacitance is either the external pin capacitance when the `-pin_load` parameter is specified, or the external wire capacitance when the `-wire_load` parameter is specified, or the total external capacitance when neither option is specified.

`-subtract_pin_load`	This parameter can be used only if nets are specified in the object list. When this parameter is specified, the total net capacitance value will be the same as set by the `set_load` *capacitance_value object_list* command. When this parameter is not specified, the asserted load value will only change the wire capacitance value from the SPEF. The total net capacitance value will be the `set_load` capacitance value plus the pin capacitance from the library file.
`-wire_load`	Interprets the specified capacitance as the external wire capacitance of the port. Use this option only for ports. Use this option with the *`port_net_list`* argument if the list contains only ports.

Examples

Here are some examples.

- The following command sets an external minimum pin capacitance of 1.5 units on the `in1 in2` ports:
  ```
  set_load 1.5 -min -pin_load [get_ports {in1 in2}]
  ```
- The following command sets an external minimum wire capacitance of 1.2 units on the `in1 in2` ports:
  ```
  set_load 1.2 -min -wire_load [get_ports {in1 in2}]
  ```
- The following command sets an external maximum pin capacitance of 2.6 units on the `in1 in2` ports:
  ```
  set_load 2.6 -max -pin_load [get_ports {in1 in2}]
  ```
- The following command sets an external maximum wire capacitance of 3.0 units on the `in1 in2` ports:
  ```
  set_load 3.0 -max -wire_load [get_ports {in1 in2}]
  ```
- The following command reports the capacitances set on the ports, as shown in the example below:
  ```
  report_ports -pin [get_ports {in1}]
  ```

  ```
  report_ports -pin [get_ports {in1}]

  -------------------------------------------------------------
  Early Late
  -------------------------------------------------------------
  Pin   Dir Assertion       Clock   Rise   Fall  Rise  Fall
  Name                      Name
  -------------------------------------------------------------
  in1   IN drive_resistance *(D)(P) 2.000  2.000  2.000  2.000
  -------------------------------------------------------------
  -----------------------------------------------------
  Pin    Dir   Assertion     Value
  Name
  -----------------------------------------------------
  in1   IN    port_cap      ( 1.500 : : 2.600 )
  in1   IN    port_wire_cap ( 1.200 : : 3.000 )
  -----------------------------------------------------
  ```

Related commands

- all_outputs
- report_ports
- report_timing
- set_drive
- set_resistance

set_logic_one

```
set_logic_one
-help
pin_or_port_list
```

Sets ports or pins to logic 1 in the design.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `set_logic_one` parameter. For a detailed description of the command and all of its parameters, use the man command: `man set_logic_one`.
`pin_or_port_list`	Specifies the list of the ports or pins that are to be set to logic 1.

Examples

- The following command sets the specified port to logic 1:
  ```
  set_logic_one [get_ports {din[1]}]
  ```

Related Commands

- `get_ports`

set_logic_zero

```
set_logic_zero
-help
pin_or_port_list
```

Sets ports or pins to logic 0 in the design.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `set_logic_zero` parameter. For a detailed description of the command and all of its parameters, use the man command: `man set_logic_zero`.
`pin_or_port_list`	Specifies the list of the ports or pins that are to be set to logic 0.

Examples

- The following command sets the specified port to logic 0:
  ```
  set_logic_zero [get_ports {din[0]}]
  ```

Related Commands

- `get_ports`

set_max_capacitance

```
set_max_capacitance
[-help]
limit
obj_list
[-clock_path]
[-data_path]
[-fall]
[-override]
[-rise]
```

Sets the maximum capacitance limit on the specified ports of the top cell, the specified modules, or clock waveforms.

When set on a port, the total capacitance of the net connected to that port must be less than or equal to the specified limit. Maximum capacitance limits also come from library objects, which is the default maximum capacitance limit of libraries and the maximum capacitance limit on library cell pins. Pins in the design inherit the limits from libraries.

Note: The limit set on the top cell applies to its ports. Thus, if maximum capacitance limits are set on the top cell and its ports, the most constraining limit will apply to the ports.

When limits are set on a clock waveform, you can use `-clock_path`, `-data_path`, `-rise`, and `-fall` parameters to provide more specifications.

The `set_max_capacitance` constraint on clock objects will be honored only by the `report_constraint` command for DRV violations. The constraint will not be used in optimizations.

Parameters

`-help`	Prints out the command usage.
`-clock_path`	Applies the max capacitance constraint only to pins in the clock network of the clocks defined in the object list.
`-data_path`	Applies the max capacitance constraint only to data path pins, where the path is launched by the clocks defined in the object list.
`-fall`	Applies the max capacitance constraint only to falling capacitance.
`-rise`	Applies the max capacitance constraint only to rising capacitance.
`limit`	Specifies the value for the minimum capacitance limit. The limit must be greater than or equal to zero. The limit is in capacitance units from the target technology library.
`obj_list`	Specifies a list of objects - ports, instance pins, library cell pins, modules, and/or clock waveforms.
`-override`	Overrides the maximum capacitance limit from other sources, such as, library or constraints. This parameter can be used to force a specific constraint value for a specified list of instance pins/port/library cell pins - this overrides the maximum/minimum values set prior to this setting. To enable this feature, make the following setting: `set timing_constraint_enable_drv_limit_override true`

Examples

Here are some examples.

- The following example sets a maximum capacitance limit of `1.2` units on all the ports of the top cell and a maximum capacitance limit of `0.5` units on the port named `d_out`:

  ```
  set_max_capacitance 1.2 [get_ports *]
  set_max_capacitance 0.5 d_out
  ```

- The following example uses the `report_ports` command to show the maximum capacitance limit set on ports:

  ```
  set_max_capacitance 0.9 [get_ports *]
  report_ports -type port_cap_limit -pin [get_ports *]
  ```

  The following report is displayed:

  ```
  ----------------------------------------
  Pin     Dir  Assertion       Value
  Name
  ----------------------------------------
  CLK2    IN   port_cap_limit  ( : 0.900 )
  CLK1    IN   port_cap_limit  ( : 0.900 )
  in2     IN   port_cap_limit  ( : 0.900 )
  in1     IN   port_cap_limit  ( : 0.900 )
  ```

```
out2    OUT  port_cap_limit  ( : 0.900 )
out1    OUT  port_cap_limit  ( : 0.900 )
----------------------------------------
```

Related Commands

- get_ports
- report_ports
- set_max_transition
- set_max_fanout
- report_constraint

set_max_delay

```
set_max_delay
value
[-combinational_from_to | -ignore_clock_latency]
[{-from |-rise_from | -fall_from} from_list]
[{-through | -rise_through | -fall_through} through_list]
[{-to | -rise_to | -fall_to} to_list]
[-comment string]
[-rise]
[-fall]
```

Specifies a maximum delay for a timing path. The `set_max_delay` command takes precedence over other path exceptions specified for the same path, except for the `set_false_path` command.

Use either the `-from`, `-to`, or `-through` option with the `set_max_delay` command to identify the paths. Specifying both the `-from` and `-to` options identifies a path with specific starting and ending points.

When you specify a sequential cell name for starting point of the timing path, the valid path starts from the clk pin. This means the latency and delay from clk to output Q is also included in the delay. For path segmentation, you can specify the path to start from Q output of the register by adding /Q after the cell name. In this case, path from clk to Q is not considered in the calculation.

Use only one `-from`, `-rise_from`, or `-fall_from`, `-to`, `-rise_to`, or `-fall_to` parameter per command. You can use multiple `-through` parameters in the same command.

Parameters

`-combinational_from_to`	Defines a path defined by -from/to pins as combinational. When this parameter is used, the path between the -`from`/-`to` pins is treated as purely combinational, that is, both the source/capture clock latencies are excluded from slack computation. The presence of -`from`/`-to` pins is required for this option to take effect; the -`from`/`-to` pins can be sequential or combinational start/end pins. The `-combinational_from_to` parameter will be deprecated in a future release. This parameter will continue to work in the current release, but for compatibility with future releases you should migrate your scripts to use the `-ignore_clock_latency` parameter. You can use the following attributes to match the semantics between the `-ignore_clock_latency` and `-combinational_from_to` parameters: • `timing_constraint_path_delay_exclude_check_delay_from_ignore_clock_latency` • `timing_constraint_path_delay_exclude_io_delay_from_ignore_clock_latency`
`-comment string`	Allows you to insert comments. The comments are read but are ignored with a one-time warning message. They are not used during timing analysis reporting nor are they written out during saving of constraints.

`-fall`	Apply the assertion at the falling edge of the signal on the path endpoints.
`-from |-rise_from |-fall_from` *from_list*	Specifies a list of timing paths start points. The *from_list* argument can contain either object IDs or hierarchical names relative to the current module or clock names. The pins can be on intermediate hierarchical boundaries. The *from_list* argument can be a collection. The valid start points are: • Input port • Bidirectional port • Clock pin of sequential cell Clock pin of sequential cell is one of the following: • Structural clock pin that is part of latch or flip-flop. • Inferred clock pin in cells like macro cells. A pin is an inferred clock pin if it is associated with one of the following: • Reference end of setup check • Reference end of hold check • Reference end of removal check • Reference end of recovery check • Minimum period check • Minimum pulse width check • Pin associated with `set_input_delay` Specify one of the following:
	`-from`: Applies the maximum delay constraint to all paths originating from the *from_list* list. If a clock is specified as the `-from` object, all primary inputs and registers clocked by that clock are used as start points. By default, the `-from` option applies the constraint to both the rising and the falling edges.
	`-rise_from`: Applies the maximum delay constraint at the rising edge of the signal on the from pins.
	`-fall_from`: Applies the maximum delay constraint at the falling edge of the signal on the from pins.

`-ignore_clock_latency`	When this parameter is specified, the software does not count the launch/capture clock latency delays into the timing path delay. The `-ignore_clock_latency` parameter is an industry-standard construct introduced in SDC 2.1 to control the handling of clock-related timing with respect to the `set_max_delay`/`set_min_delay` constraints.
`-rise`	Applies the assertion at the rising edge of the signal on the path endpoints.
`-through | -rise_through | -fall_through` *`through_list`*	Specifies a list of timing paths through points. The *`through_list`* argument can contain either object IDs or hierarchical names relative to the current module or clock names. The pins can be on intermediate hierarchical boundaries. The *`through_list`* argument can be a collection. Specify one of the following:
	`-through`: When you use this parameter without using the `-from` or `-to` options, all paths that go through the specified pins have the specified delay. When you use it with both the `-from` and `-to` options, any path starting at any pin on the `-from` list *and* going through any point on the `-through` pin list *and* going to any point on the `-to` list is affected by the constraint. By default, the `-through` option applies the constraint to both the rising and the falling edges.
	`-rise_through`: Applies the assertion at the rising edge of the signal on the through pins.
	`-fall_through`: Applies the assertion at the falling edge of the signal on the through pins.

`-to| -rise_to | -fall_to` *to_list*	Specifies a list of timing paths end points. The *to_list* argument can contain either object IDs or hierarchical names relative to the current module or clock names. The pins can be on intermediate hierarchical boundaries. The *to_list* argument can be a collection. The valid end points are: • Output port • Bidirectional port • Data pin of sequential cell Data pin of sequential cell is one of the following: • Structural data pin that is part of latch or flip-flop. • Inferred data pin in cells like macro cells. A pin is an inferred data pin if it is associated with one of the following: • Signal end of setup check • Signal end of hold check • Signal end of non-sequential setup check • Signal end of non-sequential hold check • Signal end of removal check • Signal end of recovery check • Pin associated with `set_external_delay` • Data pin of clock gating cell Select from one of the following:
	`-to`: Applies the maximum delay constraint to all paths ending in the `-to` pin list. If a clock is specified as the `-to` object, all primary outputs and registers clocked by that clock are used as endpoint. By default, the `-to` option applies the constraint to both the rising and the falling edges.
	`-rise_to`: Applies the maximum delay constraint at the rising edge of the signal on the to pins.
	`-fall_to`: Applies the maximum delay constraint at the falling edge of the signal on the to pins.
value	Specifies the maximum delay value.

Example

- The following command specifies that all paths from `in1` to `out1` must have delays less than `10` units:

```
set_max_delay 10 -from in1 -to out1
```

Related Commands

- create_clock
- set_false_path
- set_input_delay
- set_min_delay
- set_multicycle_path
- set_output_delay
- reset_path_exception

set_max_fanout

```
set_max_fanout
[-help]
[-override]
limit
obj_list
```

Sets the maximum fanout load limit constraint on the specified input ports of the top cell and the specified modules.

When set on a port, the fanout load of the net connected to the port must be less than or equal to the specified limit value to meet the constraint. The fanout load of a net is the sum of the fanout loads of the pins driven by the net. When set on a module, the limit applies to all the nets of that module. Libraries and library cell pins may also have the maximum fanout load limit. In this case the most constraining limit is applied to the nets.

Use the `report_ports -type fanout_load_limit` command to view the maximum fanout load limits set on ports of the top cell.

Fanout load limit can be asserted on input ports, output pins, bidirectional pins, or ports and library pins. You cannot apply the constraint on input pins and output ports.

Parameters

`-help`	Prints out the command usage.
limit	Specifies the value of the maximum fanout load limit constraint. This is a float number and it must be greater than or equal to zero.
obj_list	Specifies a list of objects - ports, instance pins, modules, or library cell pins.
`-override`	Overrides the maximum fanout limit from other sources, such as, library or constraints. This parameter can be used to force a specific constraint value for a specified list of instance pins/port/library cell pins - this overrides the maximum/minimum values set prior to this setting. To enable this feature, make the following setting: `set timing_constraint_enable_drv_limit_override true`

Examples

- The following example sets a maximum fanout load limit of `5.0` units on all the ports of the top cell:

  ```
  set_max_fanout 5.0 [get_ports *]
  ```

- The following example shows a report of the maximum fanout load limit set on ports using the `report_ports` command, as shown in the following example:

  ```
  set_max_fanout 5.0 [get_ports *] report_ports -type fanout_load_limit -pin [get_ports *]
  ```

Maximum Fanout Load Limit Set On Ports Report

```
+-----------------------------------------------------------+
|       Pin Name         |  Dir  |     Assertion     | Value |
|------------------------+-------+-------------------+-------|
|       tdigit[7]        |  OUT  | fanout_load_limit | 5.000 |
|       tdigit[6]        |  OUT  | fanout_load_limit | 5.000 |
|       tdigit[5]        |  OUT  | fanout_load_limit | 5.000 |
|       tdigit[4]        |  OUT  | fanout_load_limit | 5.000 |
|       tdigit[3]        |  OUT  | fanout_load_limit | 5.000 |
|       tdigit[2]        |  OUT  | fanout_load_limit | 5.000 |
|       tdigit[1]        |  OUT  | fanout_load_limit | 5.000 |
|       tdigit[0]        |  OUT  | fanout_load_limit | 5.000 |
|      tdigit_flag       |  OUT  | fanout_load_limit | 5.000 |
|         reset          |  IN   | fanout_load_limit | 5.000 |
|          int           |  IN   | fanout_load_limit | 5.000 |
| port_pad_data_out[15]  |  OUT  | fanout_load_limit | 5.000 |
| port_pad_data_out[14]  |  OUT  | fanout_load_limit | 5.000 |
| port_pad_data_out[13]  |  OUT  | fanout_load_limit | 5.000 |
|        refclk          |  IN   | fanout_load_limit | 5.000 |
|         vcop           |  OUT  | fanout_load_limit | 5.000 |
|         vcom           |  OUT  | fanout_load_limit | 5.000 |
|        pllrst          |  IN   | fanout_load_limit | 5.000 |
|         ibias          |  IN   | fanout_load_limit | 5.000 |
+-----------------------------------------------------------+
```

Related Commands

- get_ports
- report_ports
- set_fanout_load
- set_max_capacitance
- set_max_transition
- report_constraint

set_max_time_borrow

```
set_max_time_borrow
value
object_list
```

Specifies the maximum time that can be borrowed by one stage from the next logic stage following a latch to meet timing constraints. Time borrowing is sometimes called cycle stealing. Time borrowing is performed automatically when performing timing analysis.

Use the `set_max_time_borrow` assertion on pins or waveforms. If the assertion is placed on a clock waveform, all latches triggered by this clock signal get the specified maximum time borrow limit. If both the clock waveform arriving at the clock pin of a latch and the clock pin itself have time borrow limit constraints, the maximum borrow limit on the latch is decided by the assertion on the latch clock pin.

Parameters

value	Specifies the maximum amount of time that can be borrowed. The minimum value allowed is `0`, which disables time borrowing. The maximum allowed is equal to the pulse width minus the setup time: Max = pulse width - setup
object_list	Specifies the list of instance pins, instances, or clocks on which the borrow value is applied. The *object_list* argument can be a collection.

Examples

```
set_max_time_borrow 2 [get_cells {i1, i2, i3}]

set_max_time_borrow 1.7 I4/g

set_max_time_borrow 2.5 [get_clocks CLK1]
```

Related Commands

- get_clocks
- get_cells
- get_pins
- reset_max_time_borrow

set_max_transition

```
set_max_transition
[-help]
[-clock_path]
[-data_path]
[-rise]
[-fall]
[-override]
transition_limit
object_list
```

Sets the maximum slew time limit (transition) on the specified ports of the top cell, and/or clock waveforms.

When set on a port, the slew time at the port must be less than or equal to the specified limit. Maximum transition limits on pins can come from library objects, which are the default maximum transition of the libraries or maximum transition on library cell pins. In this case, the most constraining limit, which is the smallest, will apply on pins and ports.

Use the `report_ports -type slew_limit` command to view the maximum transition limits set on ports of the top cell.

When limits are set on a clock waveform, you can use `-clock_path`, `-data_path`, `-rise`, and `-fall` parameters to provide more specifications.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `set_max_transition` parameter. For a detailed description of the command and all of its parameters, use the man command: `man set_max_transition`.
`-clock_path`	Applies the max transition constraint only to pins in the clock network of the clocks defined in the object list.
`-data_path`	Applies the max transition constraint only to data path pins, where the path is launched by the clocks defined in the object list.
`-fall`	Applies the max transition constraint only to falling transitions.
`object_list`	Specifies the list of objects - ports, instance pins, library cell pins, clock waveforms, or current design.
`-override`	Overrides the maximum transition limit from other sources, such as, library or constraints. This parameter can be used to force a specific constraint value for a specified list of instance pins/port/library cell pins - this overrides the maximum/minimum values set prior to this setting. To enable this feature, make the following setting: `set timing_constraint_enable_drv_limit_override true`
`-rise`	Applies the max transition constraint only to rising transitions.
`transition_limit`	Specifies the value of the maximum slew time limit (maximum transition), which must be greater than or equal to zero. The maximum slew time limit is in time units in the current session.

Examples

- The following command sets a maximum transition limit of `0.9` units on all the ports of the top cell:
  ```
  set_max_transition 0.9 [get_ports *]
  ```
- The following example shows a report of the maximum transition limit set on ports using the `report_ports` command, as shown in example:
  ```
  set_max_transition 0.9 [get_ports *]
  report_ports -type slew_limit -pin [get_ports *]
  ```

Maximum Transition Limit Set on Ports Report

Pin Name	Dir	Assertion	Value
in1	IN	slew_limit	(: 0.9000)
in2	IN	slew_limit	(: 0.9000)
out1	OUT	slew_limit	(: 0.9000)

out2	OUT	slew_limit	(: 0.9000)
out3	OUT	slew_limit	(: 0.9000)
out4	OUT	slew_limit	(: 0.9000)

- The following command sets the maximum transition limit of 0.001 units on pins in the `all_clocks` collection:
  `set_max_transition 0.001 -clock_path [all_clocks]`
- The following command sets the transitions to the specified value for your entire design:
  `set_max_transition` *`value`* `[current_design]`

Related Commands

- get_ports
- report_ports
- set_max_capacitance
- report_constraint

set_min_capacitance

```
set_min_capacitance
[-help]
[-clock_path]
[-data_path]
[-rise]
[-fall]
[-override]
capacitance_limit
obj_list
```

Sets the minimum capacitance limit on the specified ports of the top timing module and/or clock waveforms.

When set on a port, the total capacitance of the net connected to that port must be greater than or equal to the specified limit. When set on a module, the limit applies to all the nets of that module. The total capacitance of each net of the module must be greater than or equal the specified limit. The minimum capacitance limit also comes from library objects, which is the default minimum capacitance limit of libraries and minimum capacitance limit on library cell pins. Pins in the design inherit the limits from libraries. When limits are set on a module and the limit is inherited from libraries, the most constraining limit, which is the largest limit, is used.

Note: The limit set on the top timing module applies to its ports. Thus, if minimum capacitance limits are set on the top timing module and its ports, the most constraining limit will apply to the ports.

Use the `report_ports -type port_cap_limit` command to view the minimum capacitance limits set on ports.

When limits are set on a clock waveform, you can use `-clock_path`, `-data_path`, `-rise`, and `-fall` parameters to provide more specifications.

The `set_min_capacitance` constraint on clock objects will be honored only by the `report_constraint` command for DRV violations. The constraint will not be used in optimizations.

Note: This command syntax is applicable only when the flow-compatible mode is on.

Parameters

`-help`	Prints out the command usage.
`-clock_path`	Specifies all pins that are in the network of the specific clocks in the object list.
`-data_path`	Specifies all pins that are in the data paths launched by the specific clocks in the object list.
`-fall`	Specifies falling capacitance.
`-rise`	Specifies rising capacitance for all selected pins.
`capacitance_limit`	Specifies the value for the minimum capacitance limit. The limit must be greater than or equal to zero. The limit is in capacitance units from the target technology library.
`obj_list`	Specifies the list of objects - ports, instance pins, library cell pins, modules, and/or clock waveforms.
`-override`	Overrides the minimum capacitance limit from other sources, such as, library or constraints. This parameter can be used to force a specific constraint value for a specified list of instance pins/port/library cell pins - this overrides the maximum/minimum values set prior to this setting. To enable this feature, make the following setting: `set timing_constraint_enable_drv_limit_override true`

Examples

- The following example sets a minimum capacitance limit of `0.02` units on all the ports of the top timing module and a minimum capacitance limit of `0.05` units on the top timing module:
  ```
  set_min_capacitance 0.02 [get_ports *]
  set_min_capacitance 0.05 [current_design]
  ```
  The limit used for ports is `0.05`, which is the most restrictive limit set on the top timing module (`0.05`) and the limit set on ports (`0.02`).
- The following example uses the `report_port` command to show the minimum capacitance limit set on ports:
  ```
  set_min_capacitance 0.02 [get_ports *]
  report_ports -type port_cap_limit -pin [get_ports *]
  ```
- The following command sets the minimum capacitance limit of `0.001` units on pins in the `all_clocks` collection:
  ```
  set_min_capacitance 0.001 -data_path [all_clocks]
  ```

Minimum Capacitance Limit Set on Ports Report

```
+---------------------------------------+
| Report  | report_ports                |
|---------+-----------------------------|
| Options | -type port_cap_limit -pin 0x8 |
+---------+-----------------------------+
| Date    | 20040318.123320             |
| Tool    | pks_shell                   |
| Release | v5.14-s043                  |
| Version | Mar  5 2004 16:14:20        |
+---------------------------------------+
     +-----------------------------------------------+
     |  Pin  |  Dir  |   Assertion    |     Value     |
     |  Name |       |                |               |
     |-------+-------+----------------+---------------|
     |  in1  |  IN   | port_cap_limit | ( 0.0200 :  ) |
     |  in2  |  IN   | port_cap_limit | ( 0.0200 :  ) |
     |  out1 |  OUT  | port_cap_limit | ( 0.0200 :  ) |
     |  out2 |  OUT  | port_cap_limit | ( 0.0200 :  ) |
     |  out3 |  OUT  | port_cap_limit | ( 0.0200 :  ) |
     |  out4 |  OUT  | port_cap_limit | ( 0.0200 :  ) |
     +-----------------------------------------------+
```

Related Commands

- report_ports
- set_max_capacitance
- set_max_transition
- set_min_transition
- report_constraint

set_min_delay

```
set_min_delay
[-combinational_from_to | -ignore_clock_latency]
[{-from |-rise_from | -fall_from} from_list]
[{-through | -rise_through | -fall_through} through_list]
[{-to | -rise_to | -fall_to} to_list]
[-rise]
[-fall]
[-comment string]
value
```

Specifies a minimum delay for a timing path. The `set_min_delay` command takes precedence over other path exceptions specified for the same path, except for `set_false_path` command specifications.

Use either the `-from`, `-to`, or `-through` option with the `set_min_delay` command to identify the paths. Specifying both the `-from` and `-to` options identifies a path with specific starting and ending points. If a clock is specified as the `-to` object, all primary outputs and registers clocked by that clock are used as an endpoint.

When you specify a sequential cell name for starting point of the timing path, the valid path starts from the clk pin. This means the latency and delay from clk to output Q is also included in the delay. For path segmentation, you can specify the path to start from Q output of the register by adding /Q after the cell name. In this case, path from clk to Q is not considered in the calculation. For example, in the following set of commands, the valid path starts from different points:

```
set_min_delay 0.66 -from TR1/BR1 -to BLK/BR1/D

set_min_delay 0.66 -from TR1/BR1/Q -to BLK/BR1/D
```

For the first command, the valid path starts from clk, and for the second command the valid path starts from the Q output. In the second case, the path from clk to Q is not considered. For an example of the results generated for paths with and without path segmentation, see the Examples section below.

Use only one `-from`, `-rise_from`, or `-fall_from`, `-to`, `-rise_to`, or `-fall_to` parameter per command. You can use multiple `-through` parameters in the same command.

Parameters

`-combinational_from_to`	Defines a path defined by -from/to pins as combinational. When this parameter is used, the path between the -`from`/-`to` pins is treated as purely combinational, that is, both the source/capture clock latencies are excluded from slack computation. The presence of -`from`/`-to` pins is required for this option to take effect; the -`from`/`-to` pins can be sequential or combinational start/end pins. The `-combinational_from_to` parameter will be deprecated in a future release. This parameter will continue to work in the current release, but for compatibility with future releases you should migrate your scripts to use the `-ignore_clock_latency` parameter. You can use the following attributes to match the semantics between the `-ignore_clock_latency` and `-combinational_from_to` parameters: • `timing_constraint_path_delay_exclude_check_delay_from_ignore_clock_latency` • `timing_constraint_path_delay_exclude_io_delay_from_ignore_clock_latency`
`-comment` *string*	Allows you to insert comments. The comments are read but are ignored with a one-time warning message. They are not used during timing analysis reporting nor are they written out during saving of constraints.
`-fall`	Applies the assertion at the falling edge of the signal on the path endpoints.

`-from |-rise_from	-fall_from` *`from_list`*	Specifies a list of timing paths start points. The *`from_list`* argument can contain either object IDs or hierarchical names relative to the current module or clock names. The *`from_list`* argument can be a collection. The valid start points are: • Input port • Bidirectional port • Clock pin of sequential cell Clock pin of sequential cell is one of the following: • Structural clock pin that is part of latch or flip-flop. • Inferred clock pin in cells like macro cells. A pin is an inferred clock pin if it is associated with one of the following: • Reference end of setup check • Reference end of hold check • Reference end of removal check • Reference end of recovery check • Minimum period check • Minimum pulse width check • Pin associated with set_input_delay Specify one of the following:
	`-from`: Applies the minimum delay constraint to all paths originating from the *`from_list`* list. If a clock is specified as the `-from` object, all primary inputs and registers clocked by that clock are used as start points. By default, the `-from` option applies the constraints to both the rising and the falling edges.
	`-rise_from`: Applies the minimum delay constraint at the rising edge of the signal on the from pins.
	`-fall_from`: Applies the minimum delay constraint at the falling edge of the signal on the from pins.
`-ignore_clock_latency`	When this parameter is specified, the software does not count the launch/capture clock latency delays into the timing path delay. The `-ignore_clock_latency` parameter is an industry-standard construct introduced in SDC 2.1 to control the handling of clock-related timing with respect to the `set_max_delay`/`set_min_delay` constraints.

`-rise`	Applies the assertion at the rising edge of the signal on the path endpoints.
`-through | -rise_through | -fall_through` *`through_list`*	Specifies a list of timing paths through points. The *`through_list`* argument can contain either object IDs or hierarchical names relative to the current module or clock names. The pins can be on intermediate hierarchical boundaries. The *`through_list`* argument can be a collection. Specify one of the following:
	`-through`: When you use this parameter without using the `-from` or `-to` options, all paths that go through the specified pins have the specified delay. When you use it with both the `-from` and `-to` options, any path starting at any pin on the `-from` list *and* going through any point on the `-through` pin list *and* going to any point on the `-to` list is affected by the constraint. By default, the `-through` option applies the constraint to both the rising and the falling edges.
	`-rise_through: Applies the assertion at the rising edge of the signal on the through pins.`
	`-fall_through: Applies the assertion at the falling edge of the signal on the through pins.`

`-to	-rise_to	-fall_to` *`to_list`*	Specifies a list of timing paths end points. The *`to_list`* argument can contain either object IDs or hierarchical names relative to the current module or clock names. The pins can be on intermediate hierarchical boundaries. The *`pin_or_clock_list`* argument can be a collection. The valid end points are: • Output port • Bidirectional port • Data pin of sequential cell Data pin of sequential cell is one of the following: • Structural data pin that is part of latch or flip-flop. • Inferred data pin in cells like macro cells. A pin is an inferred data pin if it is associated with one of the following: • Signal end of setup check • Signal end of hold check • Signal end of non-sequential setup check • Signal end of non-sequential hold check • Signal end of removal check • Signal end of recovery check • Pin associated with `set_external_delay` • Data pin of clock gating cell Select from one of the following:
	`-to`: Applies the minimum delay constraint to all paths ending in the `-to` pin list. If a clock is specified as the `-to` object, all primary outputs and registers clocked by that clock are used as endpoint. By default, the `-to` option applies the constraint to both the rising and the falling edges.
	`-rise_to`: Applies the minimum delay constraint at the rising edge of the signal on the to pins.
	`-fall_to`: Applies the minimum delay constraint at the falling edge of the signal on the to pins.
`value`	Specifies the minimum delay value.

Examples

- The following command specifies a minimum delay of `2.5` for paths from `in1` to `out1`:

  ```
  set_min_delay 2.5 -from in1 -to out1
  ```

- The following commands show the difference in `report_timing` results when you specify a minimum delay with and without path segmentation:

  ```
  set_min_delay 9 -from dff1 -to dff2/d
  set_min_delay 9 -from dff1/q -to dff2/d
  ```

  The `report_timing` results for the first command with path segmentation are:

  ```
  report_timing -early -from dff1
  Path 1: VIOLATED Hold Check with Pin dff2/cp
  Endpoint: dff2/d (v) checked with leading edge of 'clk'
  Beginpoint: dff1/q (^) triggered by leading edge of 'clk'
  Other End Arrival Time 0.000
  + Hold -1.956
  + Path Delay 9.000
  = Required Time 7.044
   Arrival Time 6.148
   Slack Time -0.896
   Clock Rise Edge 0.000
   = Beginpoint Arrival Time 0.000
   +-------------------------------------------------------------+
   | Instance | Arc         | Cell    | Delay | Arrival  | Required |
   |          |             |         |       | Time     | Time     |
   |----------+-------------+---------+-------+----------+----------|
   |          | clk ^       |         |       | 0.000    | 0.896    |
   | dff1     | cp ^ -> q ^ | dfn3d1  | 4.250 | 4.250    | 5.146    |
   | inv3     | i ^ -> zn v | in01d1  | 1.898 | 6.148    | 7.044    |
   | dff2     | d v | dfn3d1 | 0.000 | 6.148 | 7.044    |
   +-------------------------------------------------------------+
  ```

  The report_timing results for the second command without path segmentation are:

  ```
  report_timing -early -from dff1
  Path 1: MET Hold Check with Pin dff2/cp
  Endpoint: dff2/d (v) checked with leading edge of 'clk'
  Beginpoint: dff1/q (^) triggered by leading edge of 'clk'
  Other End Arrival Time 0.000
  + Hold -1.956
  + Phase Shift 0.000
  = Required Time -1.956
   Arrival Time 6.148
  ```

```
Slack Time 8.104
Clock Rise Edge 0.000
= Beginpoint Arrival Time 0.000
+-------------------------------------------------------------+
| Instance  | Arc          | Cell    | Delay | Arrival | Required |
|           |              |         |       | Time    | Time     |
|----------+-------------+--------+-------+---------+----------|
|          | clk ^        |         |       | 0.000   | -8.104 |
| dff1     | cp ^ -> q ^  | dfn3d1  | 4.250 | 4.250   | -3.854 |
| inv3     | i ^ -> zn v  | in01d1  | 1.898 | 6.148   | -1.956 |
| dff2     | d v | dfn3d1 | 0.000   | 6.148 | -1.956  |
+-------------------------------------------------------------+
```

Related Commands

- set_input_delay
- set_output_delay
- set_false_path
- set_max_delay
- set_multicycle_path
- reset_path_exception

set_min_fanout

```
set_min_fanout
[-help]
[-override]
limit
obj_list
```

Sets the minimum fanout load limit constraint on the specified input ports of the top cell.

When set on a port, the fanout load of the net connected to the port must be greater than or equal to the specified limit value to meet the constraint. The fanout load of a net is the sum of the fanout loads of the pins driven by the net. When set on a module, the limit applies to all the nets of that module. The libraries and library cell pins may also have the minimum fanout load limit. In this case the most constraining limit is applied to the nets.

The constraint will not be used in optimizations.

Fanout load limit can be asserted on input ports, output pins, bidirectional pins, or ports and library pins. The constraint cannot be applied on input pins and output ports.

Parameters

`-help`	Prints out the command usage.
limit	Specifies the value of the minimum fanout load limit constraint. The specified value must be greater than or equal to zero.
obj_list	Specifies a list of objects - ports, instance pins, modules, or library cell pins.
`-override`	Overrides the minimum fanout load limit from other sources, such as, library or constraints. This parameter can be used to force a specific constraint value for a specified list of instance pins/port/library cell pins - this overrides the maximum/minimum values set prior to this setting. To enable this feature, make the following setting: `set timing_constraint_enable_drv_limit_override true`

Examples

- The following example sets a minimum fanout load limit of `0.001` units on all the ports of the top cell:
  set_min_fanout 0.001 [get_ports *]
- The following example shows a report of the minimum fanout load limit set on ports using the `report_ports` command, as shown in the example below:
  ```
  set_min_fanout 5.0 [get_ports *] report_ports -type fanout_load_limit -pin [get_ports *]
  ```

  Example: Minimum Fanout Load Limit Set On Ports Report

  ```
  > report_ports -type fanout_load_limit -pin [get_ports *]
  ----------------------------------------
  ```

```
Pin      Dir  Assertion           Value
Name
-----------------------------------------
CLK2     IN   fanout_load_limit   5.000
CLK1     IN   fanout_load_limit   5.000
in2      IN   fanout_load_limit   5.000
in1      IN   fanout_load_limit   5.000
-----------------------------------------
```

Related Commands

- get_ports
- report_ports
- set_fanout_load
- set_max_capacitance
- set_max_transition
- report_constraint

set_min_pulse_width

```
set_min_pulse_width
[-help]
[-low]
[-high]
[-waveform_aware_type {absolute | source_width_ratio}]
value
object_list
```

Selects the most restrictive value when multiple minimum pulse width checks are set.

Parameters

`-help`	Prints out the command usage.
`-high`	Applies the minimum pulse width check to high signal levels only. If neither the `-low` or `-high` option is specified, the value is applied to both low and high signal levels.
list	Specifies a list of pins or clock waveforms to which the minimum pulse width check is to be applied. If the list is not specified, the minimum pulse width check applies to all the pins in the clock tree of the current design. If a clock waveform is specified, all the pins driven by the clock are affected.
`-low`	Applies the minimum pulse width check to low signal levels only. If neither the `-low` or `-high` option is specified, the value is applied to both low and high signal levels.
object_list	Specifies list of objects: pins, ports, or clocks.
value	Specifies a non-negative floating minimum pulse width check value. No default value is assumed for minimum pulse width checks.
`-waveform_aware_type {absolute | source_width_ratio}`	Specifies the type of waveform-aware pulse width check to apply as `absolute` or `source_width_ratio`.

Examples

- The following command sets a minimum pulse width, both low and high, check of `1.0` to the clock network of the current design:
  ```
  set_min_pulse_width 1.0
  ```
- The following command sets a minimum pulse width low check of `2.0` to all the pins clocked by the `CLK1` clock:

```
set_min_pulse_width -low 2.0 [get_clocks CLK1]
```

- The following command sets a minimum pulse width `-high` check of `1.0` to the `CP` clock pin of register `ff1`, as shown in the following example:

```
set_min_pulse_width 0.5 u55/CK -low
set_min_pulse_width 1.0 u55/CK -high
report_min_pulse_width -pin u55/CK

------------------------------------------------------
SEQUENTIAL CLOCK PULSE WIDTH
------------------------------------------------------
Pin                 Required     Actual Pulse    Slack
                    Pulse Width  Width
------------------------------------------------------
u55/CK (low)        0.500        4.200           3.700
u55/CK (high)       1.000        5.800           4.800
------------------------------------------------------

> report_timing -check_type pulse_width -to u55/CK

Path 1: MET User PulseWidth Check with Pin u55/CK
Ending Clock Edge: u55/CK (^) checked with leading edge of 'CLK4'
Beginning Clock Edge: u55/CK (v) triggered by trailing edge of 'CLK4'
Path Groups: {CLK4}
Other End Arrival Time 0.152
- PulseWidth 0.500
+ Phase Shift 10.000
= Required Time 9.651
- Arrival Time 5.951
= Slack Time 3.700
Clock Fall Edge 6.000
+ Clock Network Latency (Prop) -0.049
= Beginpoint Arrival Time 5.951
-------------------------------------------------
Load   Slew    Delay  Arrival  Cell  Arc  Pin
                      Time
-------------------------------------------------
0.015  0.019   5.951  DFF      CK    v    u55/CK ->
-------------------------------------------------
```

Related Commands

- reset_min_pulse_width
- report_analysis_coverage
- report_min_pulse_width
- report_timing

set_min_transition

```
set_min_transition
[-help]
[-clock_path]
[-data_path]
[-rise]
[-fall]
[-override]
limit
obj_list
```

ⓘ This command syntax is applicable only when the flow-compatible mode is on.

Sets the minimum slew time limit (transition) on the specified ports of the top timing module and/or clock waveforms.

When set on a port, the slew time at the port must be greater than or equal to the specified limit. When set on a module, the limit applies to all the pins in the module and its ports if the module is the top timing module. The minimum transition limits on pins can come from library objects, which are the default minimum transition of the libraries or minimum transition on library cell pins. In this case, the most constraining limit, which is the largest, will apply on pins and ports.

Use the `report_ports -type slew_limit` command to view the minimum transition limits set on ports of the top timing module.

When limits are set on a clock waveform, you can use `-clock_path`, `-data_path`, `-rise`, and `-fall` parameters to provide more specifications.

The `set_min_transition` constraint on clock objects will be honored only by the `report_constraint` command for DRV violations. The constraint will not be used in optimizations.

Parameters

`-help`	Prints out the command usage.
`-clock_path`	Specifies all pins that are in the network of the specific clocks in the object list.
`-data_path`	Specifies all pins that are in the data paths launched by the specific clocks in the object list.
`-fall`	Specifies falling transition.
`obj_list`	Specifies the list of objects - ports, instance pins, library cell pins, modules, and/or clock waveforms.
`-override`	Overrides the minimum transition limit from other sources, such as, library or constraints. This parameter can be used to force a specific constraint value for a specified list of instance pins/port/library cell pins - this overrides the maximum/minimum values set prior to this setting. To enable this feature, make the following setting: `set timing_constraint_enable_drv_limit_override true`
`-rise`	Specifies rising transition for all selected pins.
`limit`	Specifies the value of the minimum slew time limit (minimum transition). This value must be less than or equal to zero. The minimum slew time limit is in time units from the target technology library.

Examples

- The following example sets a minimum transition limit of 0.9 units on all the ports of the top timing module, and a minimum transition limit of 0.7 units on the top timing module:

  ```
  > set_min_transition 0.9 [get_ports *]
  > set_min_transition 0.7 [current_design]
  ```

  The minimum limit that applies to ports is 0.9, which is the largest constraints value.

- The following example shows a report of the minimum transition limit set on ports using the `report_ports` command, as shown in the example below:

  ```
  > set_min_transition 0.9 [get_ports *]
  > report_ports -type slew_limit -pin [get_ports *]
  ```

  The following report is displayed:

  ```
  ------------------------------------------
  Pin     Dir    Assertion     Value
  Name
  ------------------------------------------
  CLK2    IN     slew_limit   ( 0.900 : )
  CLK1    IN     slew_limit   ( 0.900 : )
  in2     IN     slew_limit   ( 0.900 : )
  ```

```
in1    IN    slew_limit  ( 0.900 : )
out2   OUT   slew_limit  ( 0.900 : )
out1   OUT   slew_limit  ( 0.900 : )
-----------------------------------------

> report_constraint -drv_violation_type min_transition

Check type : min_transition
---------------------------
Pin : CLK4

Transition Time: 0.003
- min_transition: 0.900
-------------------------------------------------------
slack : -0.897 ( VIOLATED )
```

Related Commands

- set_max_transition
- reset_min_transition
- report_ports
- report_constraint

set_mode

```
set_mode
[-type cell]
mode_list
inst_list
```

Specifies the Liberty timing library modes to be active for a specific instance.

A library mode is a subset of active timing arcs that is defined in the timing library. Library modes are similar to conditional timing arcs, but do not require the application of constants to set. Instead of changing a cell's mode by applying constants to its input pins, you can use the `set_mode` command to refer to sets of active arcs by name (such as `READ` or `WRITE`, for a memory cell).

By default, all library modes in the timing library are active. If you specify one or more library modes using the `set_mode` command, only the timing arcs consistent with the specified mode(s) are active for the instance. All other library modes become inactive.

To reset the library modes for an instance, use the reset_mode command.

Parameters

inst_list	Specifies the instances to which to apply the library mode.
mode_list	Specifies the name of the library mode value defined in the timing library. You can specify one or more mode values, but only one mode value per defined mode group in the Liberty library.
-type cell	Sets the library mode at the instance level. **Note:** Setting the library mode at the instance level is the only form of `set_mode` that is currently supported. This parameter exists to support constraints coming from external sources.

Examples

- The following command specifies the `test_mode` library mode active for instances of b1.

  ```
  set_mode test_mode [get_cells b1]
  ```

  The following command reports the status of all library modes associated with b1:

  ```
  tempus > report_mode b1
  -------------------------------
  Mode Group test_mode of b1
  -------------------------------
  Mode     Status     Condition
  Name
  -------------------------------
  test_B   Inactive   !(Y)
  test_A   ACTIVE      Y
  -------------------------------
  ```

  The above report shows that only the timing arcs consistent with the `test_A` library mode are active for instances of b1 - and `test_B` is inactive.

- If you specify the `report_inactive_arcs` command, the report shows the disabled arcs for the b1 instance are from the `test_mode` library mode:

  ```
  ------------------------------------------------------
  Flags :  const propagated constant
           snipped loop snipped arcs
           disable set_disable_timing
           disable_clock_gating set_disable_clock_gating
           library_disable set_disable_cell_timing
           Missing_Phase No Arc Phase Data
  From     To       DisableType   ArcType                 Reason
  ------------------------------------------------------------------
  u8/CK ^   u8/D ^   const        hold_rising_clk_rise    u8/D = 1
  u8/CK ^   u8/D v   const        hold_falling_clk_rise   u8/D = 1
  u8/CK ^   u8/D ^   const        setup_rising_clk_rise   u8/D = 1
  u8/CK ^   u8/D v   const        setup_falling_clk_rise  u8/D = 1
  ------------------------------------------------------------------
  ```

Related Commands

- reset_mode
- report_mode
- report_inactive_arcs
- get_cells

set_multicycle_path

```
set_multicycle_path
[-help]
number_of_cycles
[-comment string]
[-fall]
[-fall_through through_list]
[-hold]
[-rise]
[-rise_through through_list]
[-setup]
[-through through_list]
[-start | -end]
[-from from_list | -rise_from from_list | -fall_from from_list]
[-to to_list] | -rise_to to_list | -fall_to to_list]]
```

Specifies multicycle paths between specific timing paths in the design or between clock domains. Multicycle paths are timing paths that require more than one clock period for execution. By default, all paths are considered as one cycle paths for setup and hold check evaluation. The `set_multicycle_path` command lets you specify additional cycles for the paths.

Parameters

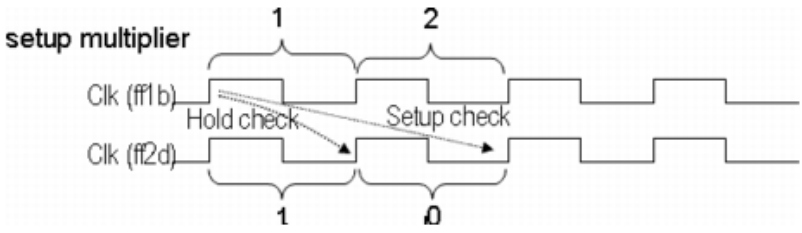
`-help`	Prints out the command usage.
`number_of_cycles`	Specifies the number of cycles allowed for the path.
`-comment` *`string`*	Allows you to insert comments. The comments are read but are ignored with a one-time warning message. They are not used during timing analysis reporting nor are they written out during saving of constraints.
`-fall`	Applies the assertion at the falling edge of the signal on the path endpoints.
`-fall_through` *`through_list`*	Applies the assertion at the falling edge of the signal on the through pins.

`-from` *`from_list`* `| -rise_from` *`from_list`* `| -fall_from` *`from_list`*	Specifies the clocks, ports, pins, or sequential instances that are at the start of the multicycle paths. The port direction can be either input or inout. Pins must be valid path startpoints such as sequential instance clock pins, latch data pins or pins on which input delays have been specified. Pins can be on intermediate hierarchical boundaries. The *`from_list`* variable can contain either collection IDs or object names. If the `-from` option is used without the `-to` option, all paths originating from the `from_list` are multicycle paths. Use either the `-from`, `-to`, or `-through` option with the `set_multicycle_path` command to identify the paths. Specifying both the `-from` and `-to` options identifies a path with specific starting and ending points. Use only one `-from`, `-rise_from`, or `-fall_from` option per command.
`-hold	-setup`	Specifies whether the multicycle adjustment applies to setup/hold check. The `-setup` parameter applies multicycles to setup check. The hold check is affected but the hold multiplier does not change. The `-hold` parameter applies multicycles to hold check only. *Default:* If you do not specify `-setup` or `-hold` parameter, the software considers the specified multicycle value for setup check and uses 0 for the hold check.
`-rise`	Applies the assertion at the rising edge of the signal on the path endpoints.
`-rise_through` *`through_list`*	Applies the assertion at the rising edge of the signal on the through pins.
`-start	-end`	Specifies whether the multicycle adjustment is to be applied relative to the starting clock edge for the path or relative to the ending clock edge for the path. Use these options for multiple clocks designs. By default, the setup check is relative to the end clock and the hold check is relative to the start clock. **Note:** The `-start` and `-end` parameters are mutually exclusive; you cannot specify them together. You can set the `timing_enable_multicycle_data_check_compatibility` attribute to `true` to enable checking for SDC compatible data checks. When the `-start` parameter is specified, the parameter reverses the reference clock by one clock cycle.

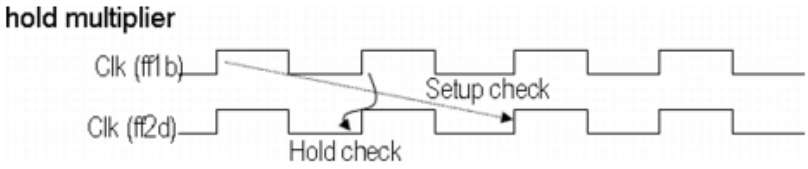
`-through` *`through_list`*	Specifies the ports, pins, instances, or nets that the path goes through. The pins can be on intermediate hierarchical boundaries. The *`through_list`* variable can contain either collection IDs or object names. When used without the `-from` or `-to` option, all paths that go through the `-through` pins are multicycle paths. When used with both the `-from` and `-to` options, any path starting at any pin on the `-from` list and going through any point on the `-through` pin list and going to any point on the `-to` list is affected by the assertion. You can use only one `-from` and `-to` option, but you can specify several `-through` options in the same command. By default, the `-through` option applies the assertion to both the rising and the falling edges. Using the `-rise_through` option applies the assertion at the rising edge of the signal on the through pins. Using the `-fall_through` option applies the assertion at the falling edge of the signal on the through pins.
`-to` *`to_list`*`]` `|` `-rise_to` *`to_list`* `|` `-fall_to` *`to_list`*	Specifies the clocks, ports, pins, or sequential instances that are at the end of the multicycle paths. The port direction can be either output or inout. Pins must be valid path end-points such as sequential instance data pins or pins on which output delays have been specified. Pins can be on intermediate hierarchical boundaries. The *`to_list`* variable can contain either collection IDs or object names. If the `-to` option is used without the `-from` option, all paths ending in the `-to` pin list are multicycle paths. Use only one `-to` option per command. Use only one `-to`, `-rise_to`, or `-fall_to` option per command.

Examples

- The following command sets all paths between ff1b + ff2d to two cycles for setup:
  ```
  set_multicycle_path -setup 2 -from {ff1b} -to {ff2d}
  ```



- The following command moves hold check to start clock preceding edge:
  ```
  set_multicycle_path -hold 1 -from {ff1b} -to {ff2d}
  ```



- The following commands show setup multiplier values from -1 to 3:

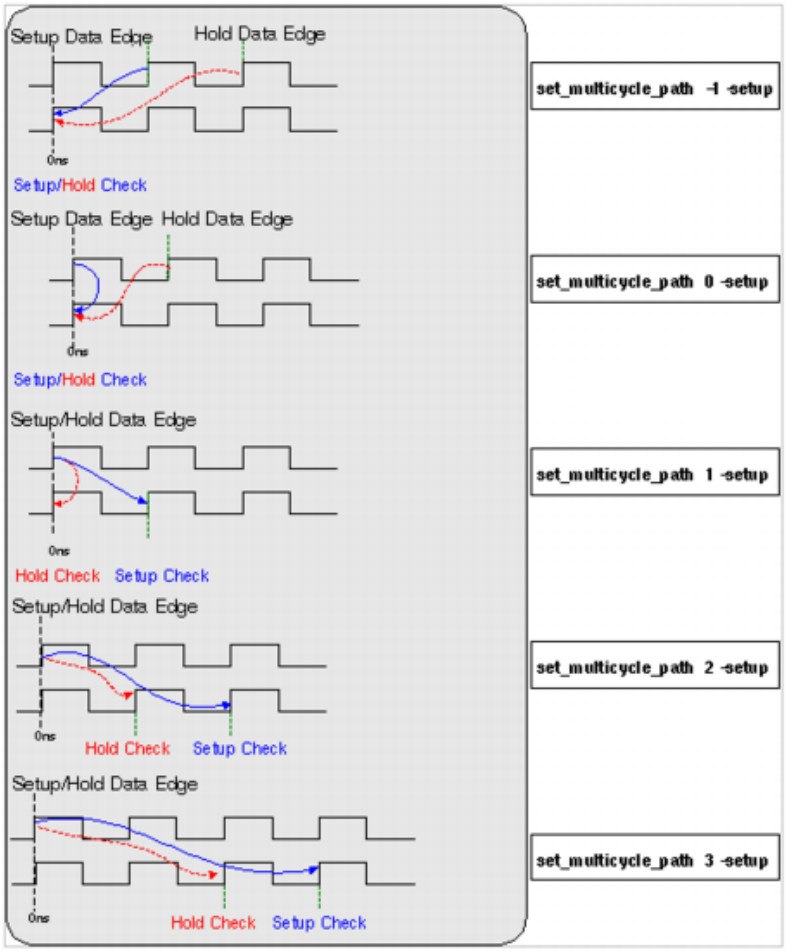
```
set_multicycle_path -1 -setup

set_multicycle_path 0 -setup

set_multicycle_path 1 -setup

set_multicycle_path 2 -setup

set_multicycle_path 3 -setup
```



- The following commands show hold multiplier values from -1 to 3:

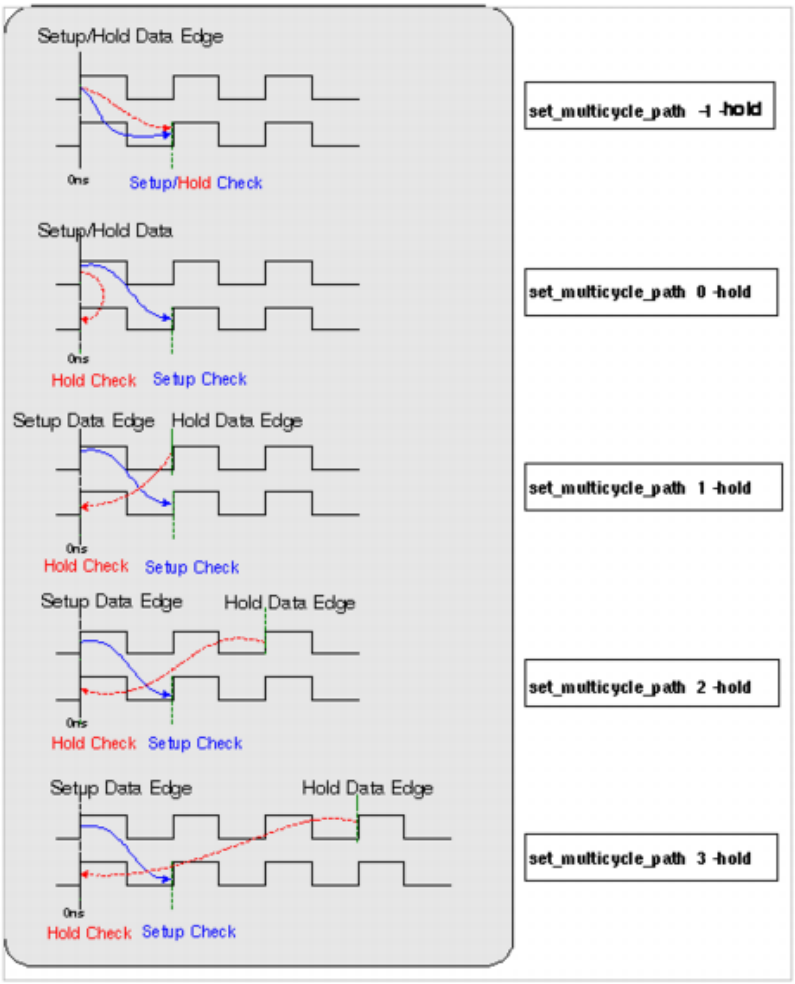
```
set_multicycle_path -1 -hold

set_multicycle_path 0 -hold

set_multicycle_path 1 -hold

set_multicycle_path 2 -hold

set_multicycle_path 3 -hold
```



Related Commands

- set_false_path
- set_max_delay
- set_min_delay
- reset_path_exception

set_normalized_driver_waveform_lib

```
set_normalized_driver_waveform_lib
[-help]
[-fall_waveform string]
[-rise_waveform string]
-ref_lib string
-libs string
```

Specifies normalized driver waveform for a library. The normalized driver waveform given in the reference library is applied to all the specified libraries.

Note: This command must be specified after the design is loaded, that is, after `set_top_module` command has been run. For this command to take effect, you should set the `timing_generate_normalized_driver_waveform` global variable to `false` before loading libraries.

Parameters

`-help`	Prints out the command usage.
`-fall_waveform string`	Specifies the name of the fall normalized driver waveform.
`-libs string`	Specifies a list of space separated libraries to which normalized driver waveform is to be applied.
`-ref_lib string`	Specifies the reference library from which normalized driver waveform is to be selected.
`-rise_waveform string`	Specifies the name of the rise normalized driver waveform.

Examples

- The following example shows that normalized driver waveform given in $libDir/libs/ref.lib will be applied to all the cells in $libDir/1.lib $testDir/2.lib $libDir/libs/3.lib:

  ```
  set_normalized_driver_waveform_lib -libs "$libDir/1.lib $testDir/2.lib $libDir/libs/3.lib" -ref_lib
  $libDir/libs/ref.lib
  ```

set_output_delay

```
set_output_delay
-clock clock_name
[-clock_fall]
[-rise] [-fall]
[-max] [-min]
[-add_delay]
[-network_latency_included]
[-source_latency_included]
[-reference_pin pin_name]
[-level_sensitive]
[-group_path group_name]
value
pin_or_port_list
```

Specifies the data required time on output ports, hierarchical ports, or internal output pins.

Parameters

`-add_delay`	Adds delay information to the existing output delay. The additional delay information can be for multiple paths relative to different clocks or clock edges leading to an input port or internal input pins. If you do not use the `-add_delay` option, the software uses the last `set_output_delay` related to a pin when there are multiple `set_output_delay` statements for the same pin.
`-clock` *clock_name*	Specifies that the delay is relative to the specified clock. You cannot use the `set_output_delay` command without the `-clock` parameter. Such constraints are accepted but are ignored during timing analysis. When you specify multiple clocks, an error occurs and the constraint is ignored.
`-clock_fall`	Specifies that the delays are with respect to the falling edge of the clock waveform. If you do not specify the `-clock_fall` option, the delay is with respect to the rising edge of the clock waveform. You cannot specify the `-clock_fall` parameter without the `-clock` parameter.
value	Specifies the value for the path delay.
`-fall`	Specifies that the output delay refers to the falling edge of the signal at specified ports or pins. If you do not specify the `-fall` option, the output delay applies to both the edges.

`-group_path` *group_name*	Specifies the group name where paths ending at the specified ports or pins are to be added. The `-group_path` parameter is not honored in multi-mode multi-corner (MMMC) designs. If specified, a warning is issued and the constraints are applied as if the `-group_path` parameter is not specified.
`-level_sensitive`	Models output delay that is captured by a positive level sensitive latch. For setup checks, the timing system checks the arrival time at the output port with the opening edge of the latch. The timing system treats any arrival time exceeding the opening edge as negative slack. For hold checks, the timing system performs the checks with respect to the closing edge of the latch. Use the `-clock` parameter to model a positive level sensitive latch, or the `-clock_fall` parameter to model a negative level sensitive latch. You cannot use the `-level_sensitive` parameter without the `-clock` parameter.
`-max`	Specifies that the delay refers to the setup analysis. If you do not specify the `-max` parameter, the output delay applies to both setup and hold analysis.
`-min`	Specifies that the delay refers to the hold analysis. If you do not specify the `-min` parameter, the output delay applies to both setup and hold analysis.
`-network_latency_included`	Specifies that the clock network latency should not be added to the output delay value. If you do not specify this option, the clock network latency of the related clock is added to the output delay value. This parameter has no effect if the specified clock is in propagated mode.
pin_or_port_list	Specifies a list of output port or pin names. The pin_or_port_list argument can be a collection.
`-reference_pin` *pin_name*	Adds source and network latency of the specified reference pin to the output delay value in propagated mode. A reference pin is a leaf pin in the fanout cone of the clock source of the clock that you specify using the `-clock` parameter. You cannot specify multiple pins for `-reference_pin` parameter. The command issues a warning and accepts the first pin. You cannot use the `-reference_pin` parameter with the `-network_latency_included` or `-source_latency_included` parameters. You cannot use the `-reference_pin` parameter to specify hierarchical pins.
`-rise`	Specifies that output delay refers to the rising edge of the signal at the specified ports or pins. If you do not specify the `-rise` option, the output delay applies to the both edges.

`-source_latency_included`	Specifies that the clock source latency should not be added to the output delay value. With different delay values for `-rise/-fall` and `-min/-max`, only the last specified combination of `-network_latency_included` and `-source_latency_included` parameters is honored. The specified combination can be one of the following: a) Either `-network_latency_included` or `-source_latency_included` b) Both the parameters c) None of these

Examples

- The following example sets the input and output delays for the bidirectional `INOUT2` port.The input signal arrives at `INOUT2` at `4.5` units after the falling edge of `CLKB`. The output signal is required at `INOUT2` at `2.5` units before the rising edge of `CLKD`:
  ```
  set_input_delay 4.5 -clock CLKB -clock_fall { INOUT2 } set_output_delay 2.5 -clock CLKD { INOUT1 }
  ```
- The following example shows how to use the `-group_path` option to add ports into a named group:
  ```
  set_output_delay 2.5 -max -clock CLK -group_path busC {busC[*]}
  ```

Related Information

- `all_outputs`
- `create_clock`
- `report_ports`
- `set_input_delay`
- `set_load`

set_pll_timing

```
set_pll_timing
[-help]
inst
-feedback_pin <feedback pin name>
-output_clock <clock name or ID>
-output_pin <output pin name>
[-reference_edge {rise | fall | both}]
-reference_pin <reference pin name>
```

This command identifies phase locked loop (PLL) insertion delays and the associated generated clock(s) that require adjustment. To remove the effect of clock tree latency, PLLs can be used to adjust the delay on the PLL output clock waveform so that the effective edges of the clock at the clock tree endpoints are aligned with the reference edges of the input clock to the PLL.

When you use the `set_pll_timing` command, the software uses the settings to automatically calculate and apply a delay offset to a PLL output clock to compensate for internal clock tree latency.

The `set_pll_timing` command works in conjunction with a generated clock definition to provide the correct clock timing at the output of the PLL. To use this command, you must first assert a generated clock on the `set_pll_timing -output_pin`. The `-source` option of the `create_generated_clock` command must also match the `-reference_pin` specified by the `set_pll_timing` command.

Parameters

`-help`	Prints out the command usage.
`inst`	Specifies the instance name or the object ID of a PLL cell, on the basis of which the reference pins, feedback pins, or the output pins will be searched.
`-feedback_pin <feedback pin name>`	Specifies the name of the PLL feedback pin.
`-output_clock <clock name or ID>`	Specifies the generated clock on the PLL output.
`-output_pin <output pin name>`	Specifies the PLL clock output pin name.
`-reference_edge {rise | fall | both}`	Defines adjustment edge(s). When this parameter is specified, the software selects the edges of the clocks at the reference and feedback pins that need to be aligned. If the rise or fall option is specified, then the clock latency from that reference edge to the related edge at the feedback pin will be used. In this case, both the rise and fall edges will use the same latency adjustment. If both of these options are specified, then the software will measure separate latency adjustments for both the rise and fall edges. *Default*: `rise`
`-reference_pin <reference pin name>`	Specifies the PLL reference pin name.

Examples

Here are some examples.

- The following example shows the defined clocks in a sample design:

  ```
  create_clock [get_ports {clk_in}] -name CLK -period 6 -waveform {0 3}
  create_generated_clock -name GCLK1 -source [get_pins {mypll/CLK}] -multiply_by 1 -master_clock
  [get_clocks {CLK}] -add [get_pins {mypll/GCLK}]
  ```

  The following `report_timing` command shows the insertion delay values at various points in the timing report:

  ```
  report_timing -from_rise mypll/GCLK -to_rise mypll/CEN -point_to_point

  Path 1:Endpoint: mypll/CEN (^) (unconstrained output)

  Beginpoint: mypll/GCLK (^) triggered by leading edge of 'GCLK1'
  ```

```
Arrival Time 0.496
Analysis View: default_analysis_view_setup
------------------------------------------------------------
Pin           Cell      Delay   Arrival   Generated Clock
                                Time      Adjustment
------------------------------------------------------------
mypll/CLK                       0.112
mypll/GCLK ->  PLL       0.133  0.246     GCLK1 Adj. = 0.000
clkTreeL0/Y    CLKBUFX1  0.139  0.385
clkTreeL1_1/Y  CLKBUFX1  0.118  0.503
mypll/CEN ->   PLL      -0.007  0.496
------------------------------------------------------------
```

From the above table, it can be seen that the rise insertion delay of clock CLK at pin mypll/CLK is 0.112ns.

Now, the following command will adjust the source latency of clock GCLK1 such that the edges at mypll/CLK and mypll/CEN are aligned i.e. the clock's rising edge should reach mypll/CEN at 0.112ns only:

```
set_pll_timing [get_cells mypll] -reference_pin CLK -feedback_pin CEN -output_pin GCLK -
output_clock [get_clocks GCLK1]
```

- The following `report_timing` command shows the source latency of GCLK clock as -0.138ns, and the clock reaches mypll/CEN pin at 0.112ns (same as on mypll/CLK):

```
report_timing -from_rise mypll/GCLK -to_rise mypll/CEN -point_to_point

Path 1:Endpoint: mypll/CEN (^) (unconstrained output)
Beginpoint: mypll/GCLK (^) triggered by leading edge of 'GCLK1'
Arrival Time 0.112
Analysis View: default_analysis_view_setup
+ Source Insertion Delay 0.112
- PLL Feedback Delay 0.250
= Beginpoint Arrival Time -0.138
------------------------------------------------------------------
Pin             Cell       Delay    Arrival    Generated Clock
                                    Time       Adjustment
------------------------------------------------------------------
mypll/GCLK ->                       -0.138    GCLK1 Pll Adj. = 0.000
clkTreeL0/Y     CLKBUFX1   0.139    0.001
clkTreeL1_1/Y   CLKBUFX1   0.118    0.120
mypll/CEN ->    PLL       -0.007    0.112
------------------------------------------------------------------
```

Related Commands

- create_generated_clock
- create_clock
- reset_pll_timing

set_propagated_clock

```
set_propagated_clock
[-help]
obj_list
```

Puts the `propagated_clock` assertion on the specified pin, port, or clock object. This will cause all clock endpoints in the fanout of the specified object to receive propagated clock timing unless there is a `set_clock_latency` with higher precedence. If there are multiple `set_clock_latency` and `set_propagated_clock` assertions between the clock waveform, clock root and clock endpoint, the closest assertion to the clock endpoint takes precedence.

There are two methods of clock propagation: depending on whether the clock is ideal or propagated.

- When a clock is ideal, the delay from a clock port to a register clock pin (network insertion delay) comes from design constraints using the `set_clock_latency` command.
- When a clock is propagated, the network insertion delay is computed from the actual gates and interconnects in the clock network.

To set the propagated_clock property on all clock waveform objects, use the following command:

```
set_propagated_clock [all_clocks]
```

When `set_propagated_clocks [all_clocks]` command is asserted, it will apply to all the currently defined clock waveform objects. This setting will not apply to clocks created later in the script. Whether all the clock endpoints receive propagated or ideal timing depends on the precedence of `set_clock_latency` in the fanout.

The `set_propagated_clock [all_clocks]` command puts the `propagated_clock` assertion on all clocks except virtual clocks. When you put the assertion on a real clock that is used to define I/O timing (with `set_input_delay -clock` or `set_output_delay -clock`), the software does not take the specified network latency into account. Instead, the software calculates the network latency, which changes the I/O timing. You can change this behavior by using the following attribute:

```
set_attribute timing_io_use_clock_network_latency always
```

You can use this command after creating the clock with the `create_clock` command.

Parameters

`-help`	Prints out the command usage.
`obj_list`	Specifies the objects on which you want to put the `propagated_clock` assertion. The *`pin_or_clock_list`* argument can be a collection. When a pin is specified, it affects the propagation mode for all the registers in the transitive fanout (TFO) of the pin.

Examples

- The following command puts the `propagated_clock` assertion on the clock waveform `CLK1`:

  ```
  set_propagated_clock CLK1
  ```

- The following command puts the `propagated_clock` assertion on all clock waveforms:
  `set_propagated_clock [all_clocks]`

Related Information

- `all_clocks`
- `get_propagated_clock`
- `report_clocks`
- `set_clock_latency`
- `set_clock_uncertainty`

set_property

```
set_property
[-obj_type {object_type}]
{obj_list}
property_name
property_value
[-quiet]
```

Sets user-defined properties on objects.

The `set_property` command needs a property to be defined using the `define_property` command.

Parameters

obj_list	Specifies a list of objects on which the property is to be applied.
-obj_type {*object_type*}	Specifies the type of object on which the property is to be defined.
property_name	Specifies the name of the property.
property_value	Specifies the value of the property.
-quiet	Specifies to exit quietly without issuing an error.

Examples

- The following command applies a user-defined property `slk` to pins `C1/A`:

  ```
  set_property [get_pins C1/A] slk 2
  ```

- The following command applies aocv_weight property to library arcs:

  ```
  set_property [get_lib_arcs -of_objects slow/INVX1] aocv_weight 8
  ```

Related Commands

- `get_property`
- `report_property`
- `define_property`
- `reset_property`

set_pulse_clock_max_transition

```
set_pulse_clock_max_transition
limit
obj_list
[-transitive_fanout]
[-rise]
[-fall]
```

Sets the maximum pulse clock transition constraint on the specified pulse generator cells, pulse generator library cells, clocks, and/or current design.

Parameters

`-fall`	Sets transition limit value for falling transition.
obj_list	Specifies a list of objects - pulse generator cells, pulse generator library cells, clocks, or current design.
`-rise`	Sets transition limit value for rising transition.
limit	Specifies the transition limit value.
`-transitive_fanout`	Specifies the transition limit value for transitive fanout.

Related Commands

- get_clocks
- get_cells
- reset_pulse_clock_max_transition

set_pulse_clock_max_width

```
set_pulse_clock_max_width
width
obj_list
[-transitive_fanout]
```

Sets the maximum pulse clock width constraint on the specified pulse generator cells, pulse generator library cells, clocks, and/or current design.

Parameters

obj_list	Specifies a list of objects - pulse generator cells, pulse generator library cells, clocks, or current design.
width	Specifies the pulse width value.
-transitive_fanout	Specifies the pulse width value for transitive fanout.

Related Commands

- get_clocks
- get_cells
- reset_pulse_clock_max_transition
- reset_pulse_clock_max_width

set_pulse_clock_min_transition

```
set_pulse_clock_min_transition
limit
obj_list
[-transitive_fanout]
[-rise]
[-fall]
```

Sets the minimum pulse clock transition constraint on the specified pulse generator cells, pulse generator library cells, clocks, and/or current design.

Parameters

`-fall`	Sets transition limit value for falling transition.
obj_list	Specifies a list of objects - pulse generator cells, pulse generator library cells, clocks, or current design.
`-rise`	Sets transition limit value for rising transition.
limit	Specifies the transition limit value.
`-transitive_fanout`	Specifies the transition limit value for transitive fanout.

Related Commands

- reset_pulse_clock_min_transition
- get_clocks
- get_cells

set_pulse_clock_min_width

```
set_pulse_clock_min_width
[-help]
pulse_width
object_list
[-transitive_fanout]
```

Sets the minimum pulse clock width constraint on the specified pulse generator cells, pulse generator library cells, clocks and/or current design.

When you set this constraint on a clock, the settings are applied to all the pulse generators that are driven by the clock. When this constraint is used for a design, the software applies the constraint to all the clock networks in the design.

The specified minimum pulse width is expected to be larger than the pulse width value. The width is defined when a clock object is created and its waveform is defined in the current design (using the `create_clock -waveform` parameter).

In case of conflicting minimum width constraints, the more restrictive constraint is used. When applied on a non-pulse generator object, or a non-clock target, this constraint has no effect.

Note: You must specify `set_interactive_constraint_modes` constraint prior to running the `set_pulse_clock_min_width` command.

Parameters

`-help`	Prints out the command usage.
`object_list`	Specifies a list of objects - pulse generator cells, pulse generator library cells, clocks, or current design.
`pulse_width`	Specifies the minimum pulse width value.
`-transitive_fanout`	Specifies the pulse width value for transitive fanout.

Examples

- The following example sets the min pulse clock width constraint of 0.3 ns on the pulse generator instance `pg_inst`:

  ```
  set_pulse_clock_min_width 0.3 pg_inst
  ```

  This constraint can be observed using the following command:
  `report_constraint pg_inst/Y -check_type pulse_clock_min_width`

  Fragmented output:

  ```
  ----------------------------------------
  Pin Required Actual Pulse Slack Clock Type

  Pulse Width Width
  ----------------------------------------
  pg_inst/Y (high)    0.300 0.366 0.066 PG
  ----------------------------------------
  ```

Related Commands

- `report_constraint`
- `reset_pulse_clock_min_width`
- `set_pulse_clock_max_width`
- `report_min_pulse_width`
- `set_interactive_constraint_modes`
- `timing_enable_uncertainty_for_pulse_width_checks`

set_resistance

```
set_resistance
-help
[-max]
[-min]
value
net_list
```

Sets the wire resistance value on the specified nets of the top cell, which is used to compute the net delay of the specified nets. The resistance that you specify using this command overrides the resistance from wire load models and SPEF.

Note: In the best-case worst-case (BC-WC) analysis mode, the minimum resistance is used to compute the early path delay and the late path delay for the minimum corner. The maximum resistance is used to compute the early path delay and the late path delay for the maximum corner.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `set_resistance` parameter. For a detailed description of the command and all of its parameters, use the man command: `man set_resistance`.
`-max`	Specifies that the asserted resistance is the maximum resistance. *Default*: If the `-max` (or `-min`) parameter is not specified, the resistance value is used as both minimum and maximum resistance.
`-min`	Specifies that the asserted resistance is the minimum resistance. *Default*: If the `-min` (or `-max`) parameter is not specified, the resistance value is used as both minimum and maximum resistance.
`net_list`	Specifies a list of nets on which the wire resistance is set.
`resistance_value`	Specifies the value for the wire resistance. This value (in units) is found in the target technology library.

Examples

- The following example set the maximum wire resistance of `2.0` units on the net `n1`:

  ```
  set_resistance -max 2.0 [get_nets n1]
  ```

- The following example set the minimum wire resistance of `1.2` units and a maximum wire resistance of `2.0` units on the net `n1`:

  ```
  set_resistance 1.2 -min [get_nets n1]
  ```

```
set_resistance 2.0 -max [get_nets n1]
```

Related Commands

- reset_resistance
- report_net
- report_timing
- set_drive
- set_load

write_annotated_transition

```
write_annotated_transition
[-help]
out_file
[-direct]
[-exclude_ports]
[-include_output_pins]
[-max]
[-min]
[-view string]
```

Writes the value of transitions on each pin to an output file. You can either set the transitions using the `set_annotated_transition` command or the software computes them internally.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `write_annotated_transition` parameter.
`-direct`	Refers the ports directly for which the annotated transition will be applied in the output SDC. This option removes the dependency of using the `get_ports` and `get_pins` commands.
`-include_output_pins`	Writes the slews for input and output pins, and the output slew for bi-directional pins. If this option is not specified, the software writes the slew for input pins and the input slew of bi-directional pins.
`-min/-max`	Specifies whether to write out minimum or maximum transition. *Default:* Both minimum and maximum transitions are written out.
`-exclude_ports`	Skips writing the transition values for the primary ports (input and output).
out_file	Specifies the name of the output file.
`-view` *string*	Writes the transitions for the specified analysis view only. You can specify this parameter only when the software is in multi-mode multi-corner timing analysis mode.

Related Commands

- `set_annotated_transition`
- `get_ports`
- `get_pins`

set_sense

```
set_sense
[-help]
pin_or_port_list
[-clocks clock_list]
{[-positive | -negative | -stop_propagation ] [-type <type_list> ]}
```

Selects which phase of the clock to filter at the specified point.

Parameters

`-help`	Prints out the command usage.
`-clocks` *`clock_list`*	Specifies a list of clocks passing through a specified point to which the constraint applies. When this parameter is not specified, the filtering of clock phase will be applicable to all the clocks passing through that specified point. This parameter must be used with a specified pin name and one of the parameters : `-positive`, `-negative`, or `-stop_propagation`.
`-negative`	Specifies that positive phase filtered clocks pass through pins in the *`pin_list`*.
`pin_or_port_list`	Specifies the list or collection of cell output pins. The unateness specified by the `set_sense` command is applied to clocks propagating through the specified point.
`-positive`	Specifies that negative phase filtered clocks pass through pins in the *`pin_list`*.
`-stop_propagation`	Stops the propagation of specified clocks and clock source data paths passing through the specified pins.

`-type type_list`	Applies clock sense to object types specified in the *`type_list`*. The type list options are - `all`, `clock`, `data`, `clock_source_data`, `default_data`. These are explained below: • `all`: Stops all phases from propagating. This is equivalent to specifying both the `clock` and `data` options. • `clock`: Stops clock phases launched by clock(s) specified in the *`clock_list`* from propagating. • `data`: Stops all data phases launched by clock(s) specified in *`clock_list`*, starting from the pins/ports specified in *`pin_or_port_list`*. The objects in the *`pin_or_port_list`* can be selected anywhere in the design - clock/data network. • `clock_source_data`: Stops only clock source data phases from propagating. The combination of `clock_source_data` and `clock` is equivalent to the existing `-stop_propagation` option. The `-type` parameter extends the capability of the `set_clocks_sense` command to selectively block data timing phases. This parameter provides a more generalized use model for selecting among all types of paths that should be filtered. The `-type` parameter is mutually exclusive with the `-positive`, `-negative`, and `-stop_propagation` options.

Examples

- The following command allows the positive phase clock CLK1 to pass through CLKBUF1/Z and filters out the negative phase clock CLK1:
  ```
  set_sense -positive -clocks CLK1 CLKBUF1/Z
  ```
- The following command stops clock phases starting from pin1:
  ```
  set_sense pin1 -type {clock}
  ```
- The following command stops data phases based only on CK1 and CK2 starting from pin1:
  ```
  set_sense pin1 -type {data} -clocks {CK1 CK2}
  ```
- The following command stops only the clock source data phases starting from pin1:
  ```
  set_sense pin1 -type {clock_source_data}
  ```
- The following command stops the clock and clock source data phases:
  ```
  set_sense pin1 -type {clock clock_source_data}
  ```
  or
  ```
  set_sense pin1 -stop_propagation
  ```

Related Commands

- reset_sense

reset_sense

```
reset_sense
[-help]
[-all | [[-clocks <clock_list>] <pin_or_port_list>]]
```

Resets clock sense constraint(s) specified using the `set_sense` command.

Parameters

`-help`	Prints out the command usage.
`-all`	Resets all the specified `clock_sense` constraints.
`-clocks` *`clock_list`*	Specifies the list of clocks for which the clock sense constraints are to be removed. The clocks must have previously been defined using the `set_sense -clocks` parameter.
`pin_or_port_list`	Specifies the list or collection of cell output pins or ports for which the clock sense constraints will be removed.

Examples

- The following command resets all the previously defined `clock_sense` constraints from a design:

  ```
  tempus > reset_sense -all
  ```

Related Commands

- `set_sense`

40

Timing Debug Commands

- analyze_paths_by_bottleneck
- analyze_paths_by_clock_domain
- analyze_paths_by_critical_false_path
- analyze_paths_by_default_path_group
- analyze_paths_by_drv
- analyze_paths_by_hier_port
- analyze_paths_by_hierarchy
- analyze_paths_by_view
- create_histogram_png
- create_path_category
- delete_path_category
- gui_gtd_highlight_timing_report
- gui_open_gtd
- read_path_categories
- read_timing_debug_report
- write_category_summary
- write_histogram_view
- write_path_categories
- write_path_list_summary
- write_text_timing_report

analyze_paths_by_bottleneck

```
analyze_paths_by_bottleneck
[-help]
{-category n | -delta_delay {worst|total}}
[-path_num m]
[-max_slack float]
[-min_slack float]
[-master master_category_name]
```

Creates categories according to the bottleneck analysis of the critical paths and fixes the violation according to the bottleneck information. Bottleneck Analysis detects the instances that are often involved in the critical paths in the design. Any change in timing for these instances impacts multiple critical paths in the design. Therefore you can use this information to change your design (floorplan, placement, constraints etc.) around these instances. The bottleneck analysis computes the number of occurrences that an instance appears in the paths that you display in the Timing Debug window, and provides the following output:

- Colors most used instance based on number of occurrences.
- Creates *n* number of path categories for the instances with the highest number of occurrences. *n* is the number you specify using the `-category` parameter.

Users can also perform bottleneck analysis of the critical paths based on incremental delay. Categories will be created based on the worst *m* paths with:

- The biggest worst `incr_delay` contribution
- The biggest total `incr_delay` contribution

The command can be applied wherever the user loads the machine readable timing report.

For more information on creating path categories, see the 'Creating Path Categories' section in the Debugging Timing Results chapter of the *Innovus User Guide* document.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `analyze_paths_by_bottleneck` parameter. For a detailed description of the command and all its parameters, use the `man` command: `man analyze_paths_by_bottleneck`
`-category` *n*	Specifies the number of categories to be created for most occurring instances. *Data_type*: int, required
`-delta_delay {worst|total}`	Specifies that the categories are created based on worst `incr_delay` contribution or total `incr_delay` contribution. *Data_type*: enum, required
`-master` *master_category_name*	Specifies that the analysis should be performed only for the specified `master` category. *Data_type*: string, optional
`-max_slack` *float*	Specifies the max slack value of the critical paths for bottleneck analysis based on `incr_delay`. *Data_type*: float, optional
`-min_slack` *float*	Specifies the min slack value of the critical paths for bottleneck analysis based on `incr_delay`. *Data_type*: float, optional
`-path_num` *m*	Specifies the number of critical paths to be analyzed when performing bottleneck analysis based on `incr_delay`. *Data_type*: int, optional

Examples

- The following command creates path categories based on bottleneck analysis of the critical paths on the instances:

  ```
  analyze_paths_by_bottleneck -category 2
  ```

  The following categories are created: `BOT_instance1` and `BOT_instance2`. You can change the number of categories.

- The following command creates path categories based on bottleneck analysis of the critical 20 paths on the worst `incr_delay`:

  ```
  analyze_paths_by_bottleneck -path_num 20 -delta_delay worst
  ```

  The following category is created: `BOT_SI_WORST`.

Related Topics

The Timing Budgeting chapter of the *Innovus User Guide* document.

analyze_paths_by_clock_domain

```
analyze_paths_by_clock_domain
[-help]
[-include_edges]
[-include_views]
[include_mode]
[-master master_category_name]
```

To find the possible timing issues in clock combinations, the command creates categories according to launch clocks - capture clock combinations.

Following categories are created:

- Paths with clock fully contained in a single domain, clk1.
- Paths with clocks starting one clock domain and ending at another, clk1-> clk2.

The command can be applied wherever the user loads the machine readable timing report.

For more information on creating path categories, see the 'Creating Path Categories' section in the Debugging Timing Results chapter of the *Innovus User Guide* document.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `analyze_paths_by_clock_domain` parameter. For a detailed description of the command and all its parameters, use the `man` command: `man analyze_paths_by_clock_domain`
`-include_edges`	Considers leading or trailing edge of the clock while defining categories. *Data_type*: bool, optional
`-include_mode`	Considers constraint mode while defining categories. *Data_type*: bool, optional
`-include_views`	Considers analysis views while defining categories. *Data_type*: bool, optional
`-master` *master_category_name*	Specifies that the analysis should be performed only for the specified master category. *Data_type*: string, optional

Examples

- The following command creates path categories based on clock domain analysis of the critical paths, and considers the edges of the clock:
  ```
  analyze_paths_by_clock_domain -include_edges
  ```
- The following command creates path categories based on clock domain analysis of the critical paths, and considers the analysis views:
  ```
  analyze_paths_by_clock_domain -include_views
  ```

Related Topics

The Timing Budgeting chapter of the *Innovus User Guide* document.

analyze_paths_by_critical_false_path

```
analyze_paths_by_critical_false_path
[-help]
-critical_false_paths
{false_path_only | exclude_false_path}
[-master master_category_name]
```

Identifies violating false paths, which are paths that appear as violations but cannot be exercised due to the structure of the netlist, and creates a category that contains them. With this command the user can, in advance, categorize the critical false paths together to analyze them.

The command can be applied wherever the user loads the machine readable timing report.

For more information on creating path categories, see the 'Creating Path Categories' section in the Debugging Timing Results chapter of the *Innovus User Guide* document.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `analyze_paths_by_critical_false_path` parameter. For a detailed description of the command and all its parameters, use the `man` command: `man analyze_paths_by_critical_false_path`
`-critical_false_paths`	Specifies which paths to analyze. • `false_path_only` Includes only false paths when creating the category. • `exclude_false_paths` Excludes false paths when creating the category. *Default*: `false_path_only` *Data_type*: enum, optional
`-master` *master_category_name*	Specifies that the analysis should be performed only for the specified master category. *Data_type*: string, optional

Examples:

- The following command creates path categories based on the critical false paths:
  `analyze_paths_by_critical_false_path -critical_false_path\ false_path_only`
- The following category is created: `default_report_false_path_only`

Related Topics

The Timing Budgeting chapter of the *Innovus User Guide* document.

analyze_paths_by_default_path_group

```
analyze_paths_by_default_path_group
[-help]
[-master master_category_name]
[-predefined]
```

Categorizes the timing paths based on basic path groups. The basic path groups are fundamental to timing analysis and help the user easily observe the timing information of different basic path groups. For instance, if user is only concerned about the paths from registers to registers, the user can analyze the paths in the category REG -> REG. The basic path groups categories are:

- Register to register
- Input to register
- Register to output
- Input to output

The following categories are created:

- `REG->REG`
- `IN->REG`
- `REG->OUT`
- `IN->OUT`

Timing debug can also convert user defined path groups into path categories. The command can be applied wherever the user loads the machine readable timing report.

Note: If your design does not have paths belonging to a type of path group, the software will not create that category.

For more information on creating path categories, see the 'Creating Path Categories' section in the Debugging Timing Results chapter of the *Innovus User Guide* document.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `analyze_paths_by_default_path_group` parameter. For a detailed description of the command and all its parameters, use the `man` command: `man analyze_paths_by_default_path_group`
`-master` *`master_category_name`*	Specifies that the analysis should be performed only for the specified master category. *Data_type*: string, optional
`-predefined`	Creates path categories for user defined path groups. *Data_type*: bool, optional

Examples

- The following command create path categories based on basic path groups:
  `analyze_paths_by_default_path_group`
  The following categories are created: REG->REG, IN->REG, REG->OUT, IN->OUT.
- The following command create path categories based on predefined path group "critical" :
  ```
  group_path -name critical -from {ff*/CLK } -to {ff*/D} #create predefined path group
  analyze_paths_by_default_path_group -predefined
  ```
  The category "critical" will be created based on the predefined path group name.

Related Topics

The Timing Budgeting chapter of the *Innovus User Guide* document.

analyze_paths_by_drv

```
analyze_paths_by_drv
[-help]
[[-load_cap_file capacitance_file] | [-generate_cap_file fileName]]
[[-load_transition_file transition_file] | [-generate_transition_file fileName]]
[-load_fanout_file fanout_file | [-generate_fanout_file fileName]]
[-master master_category_name]
```

Reads or generates a report containing max transition, max capacitance, and max fanout violations. Paths that are affected by the selected DRV type are grouped into a category. This command can be used if the user wants to debug the timing issues caused by DRV.

The command can be applied wherever the user loads the machine readable timing report.

For more information on creating path categories, see the 'Creating Path Categories' section in the Debugging Timing Results chapter of the *Innovus User Guide* document.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `analyze_paths_by_drv` parameter. For a detailed description of the command and all its parameters, use the `man` command: `man analyze_paths_by_drv`
`-load_cap_file` *capacitance_file*	
	Reads a file containing a list of paths containing max capacitance violations. *Data_type*: string, optional
`-load_fanout_file` *fanout_file*	
	Reads a file containing a list of paths containing max fanout violations. *Data_type*: string, optional
`-load_transition_file` *transition_file*	
	Reads a file containing a list of paths containing max transition violations. *Data_type*: string, optional

`-generate_cap_file` *`fileName`*	
	Generates a file containing a list of paths containing max capacitance violations. *Data_type*: string, optional
`-generate_fanout_file` *`fileName`*	
	Generates a file containing a list of paths containing max fanout violations. *Data_type*: string, optional
`-generate_transition_file` *`fileName`*	
	Generates a file containing a list of paths containing max transition violations. *Data_type*: string, optional
`-master` *`master_category_name`*	
	Specifies that the analysis should be performed only for the specified master category. *Data_type*: string, optional

Examples:

The following command generates DRV report files, then creates path categories based on the critical paths which are affected by DRV:

```
analyze_paths_by_drv -generate_cap_file capViolation.rpt

-generate_transition_file tranViolation.rpt

-generate_fanout_file fanoutViolation.rpt
```

Related Topics

The Timing Budgeting chapter of the *Innovus User Guide* document.

analyze_paths_by_hier_port

```
analyze_paths_by_hier_port
[-help]
[-generate_hier_file InstanceName_list]
[-load_hier_file file_name]
[-master master_category_name]
[-out_file]
```

Helps you to automatically group paths based on each hierarchical port of a given instance. You can use this command to perform the timing results analysis of a hierarchical design. For example, while looking at the setup/hold time at the same time of each port.

The command can be applied wherever the user loads the machine readable timing report.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `analyze_paths_by_hier_port` parameter. For a detailed description of the command and all its parameters, use the `man` command: `man analyze_paths_by_hier_port`
`-generate_hier_file` *`InstanceName_list`*	
	Generates hier ports fanin/fanout file when the database is available based on the given InstanceName_list. Then uses that file to create the categories. *Data_type*: string, optional
`-load_hier_file` *`file_name`*	
	Uses a previously saved hier ports fanin/fanout file when the database is not available. *Data_type*: string, optional
`-master` *`master_category_name`*	
	Specifies that the timing paths should be categorized only for the specified master category. *Data_type*: string, optional
`-out_file`	Specifies the generated hier ports fanin/fanout file name. *Data_type*: string, optional

Example

The following command categorizes paths based on hierarchical port of the instance `DTMF_INST/TDSP_CORE_INST`:

```
analyze_paths_by_hier_port -generate_hier_file DTMF_INST/TDSP_CORE_INST
```

Related Topics

The Timing Budgeting chapter of the *Innovus User Guide* document.

analyze_paths_by_hierarchy

```
  analyze_paths_by_hierarchy
[-master master_category_name]
{-floorplan_file floorplan_file [-include_macros]} | {-use_current_floorplan  [-all] [-
types type_list] [-port_to_port] [-port_to_macro] [-port_to_bbox] [-port_to_hinst] [-
port_to_stdcell] [-port_to_inst_group] [-macro_to_port] [-macro_to_macro] [-
macro_to_bbox] [-macro_to_hinst] [-macro_to_stdcell] [-macro_to_inst_group] [-
bbox_to_port] [-bbox_to_macro] [-blackbox_to_blackbox] [-bbox_to_hinst] [-
bbox_to_stdcell] [-bbox_to_inst_group] [-hinst_to_port] [-hinst_to_macro] [-
hinst_to_bbox] [-hinst_to_hinst] [-hinst_to_stdcell] [-hinst_to_inst_group] [-
stdcell_to_port] [-stdcell_to_macro] [-stdcell_to_bbox] [-stdcell_to_hinst] [-
stdcell_to_stdcell] [-stdcell_to_inst_group] [-inst_group_to_port] [-
inst_group_to_macro] [-inst_group_to_bbox] [-inst_group_to_hinst] [-
inst_group_to_stdcell] [-inst_group_to_inst_group]} | {-partitions partition_list | -
macros macro_list}
```

Creates categories according to the hierarchical characteristics of the paths. User can use either a user specified floorplan file or current floorplan file of the design in memory and can create categories according to the hierarchy of the design. This way, the paths can be narrowed down to small group.

Use Floorplan Plan

The software automatically extracts partition-boundary information from guides, regions, and fences in the floorplan file. Optionally you can specify option to sort paths based on the macros in the floorplan file.

- All paths fully contained in a single partition, Ptn1.
- Paths starting in one partition and ending in another partition, Ptn1–>Ptn2.
- Paths spanning several partitions, Ptn1–> Ptn2–>Ptn3.
- Top-level inputs to partitions, INPUT–>Ptn1.
- Paths from partitions to top-level outputs, Ptn1–>OUTPUT.

Use Current Floorplan

This method categorizes paths based on the current floor plan of the design in memory. It will rely on the grouped/ungrouped as a basis for what level to view the hierarchy as well as what levels to create timing categories.

Grouped/ungrouped status of an Hinst means whether or not its children are shown on the floor plan view.

It categorizes paths starting and ending on/within these hierarchical entities:

- Port
- Hard Macro (ungrouped)
- Black Box (ungrouped)
- Hinst (regular, partition, Black Blob, or Flex Model) (highest grouped)
- Stdcell (ungrouped)

For more information on creating path categories, see the 'Creating Path Categories' section in the Debugging Timing Results chapter of the *Innovus User Guide* document.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `analyze_paths_by_hierarchy` parameter. For a detailed description of the command and all its parameters, use the `man` command: `man analyze_paths_by_hierarchy`
`-all`	Creates categories of paths within all types *Default:* All types are categorized. *Data_type*: bool, optional
`-bbox_to_hinst`	Creates categories of paths from black box to hierarchical instance. *Data_type*: bool, optional
`-bbox_to_inst_group`	Creates categories of paths from black box to instance group. *Data_type*: bool, optional

`-bbox_to_port`	Creates categories of paths from black box to port. *Data_type*: bool, optional
`-bbox_to_macro`	Creates categories of paths from black box to macro. *Data_type*: bool, optional
`-blackbox_to_blackbox`	Creates categories of paths from black box to black box. *Data_type*: bool, optional
`-bbox_to_stdcell`	Creates categories of paths from black box to standard cell. *Data_type*: bool, optional
`-floorplan_file floorplan_file`	Specifies the name of the floorplan file. *Default:* Uses the current floorplan file. *Data_type*: string, required
`-hinst_to_bbox`	Creates categories of paths from hierarchical instance to black box. *Data_type*: bool, optional
`-hinst_to_port`	Creates categories of paths from hierarchical instance to port. *Data_type*: bool, optional
`-hinst_to_macro`	Creates categories of paths from hierarchical instance to macro. *Data_type*: bool, optional
`-hinst_to_hinst`	Creates categories of paths from hierarchical instance to hierarchical. *Data_type*: bool, optional
`-hinst_to_inst_group`	Creates categories of paths from hierarchical instance to instance group. *Data_type*: bool, optional
`-hinst_to_stdcell`	Creates categories of paths from hierarchical instance to standard cell. *Data_type*: bool, optional

`-include_macros`	Includes macros in addition to guides and fences for creating categories. *Default:* Macros are not considered. *Data_type*: bool, optional
`-inst_group_to_bbox`	Creates categories of paths from instance group to black box. *Data_type*: bool, optional
`-inst_group_to_hinst`	Creates categories of paths from instance group to hierarchical instance. *Data_type*: bool, optional
`-inst_group_to_inst_group`	Creates categories of paths from instance group to instance group. *Data_type*: bool, optional
`-inst_group_to_macro`	Creates categories of paths from instance group to macro. *Data_type*: bool, optional
`-inst_group_to_port`	Creates categories of paths from instance group to port. *Data_type*: bool, optional
`-inst_group_to_stdcell`	Creates categories of paths from instance group to standard cell. *Data_type*: bool, optional
`-macro_to_bbox`	Creates categories of paths from macro to black box. *Data_type*: bool, optional
`-macro_to_port`	Creates categories of paths from macro to port. *Data_type*: bool, optional
`-macro_to_macro`	Creates categories of paths from macro to macro. *Data_type*: bool, optional
`-macro_to_hinst`	Creates categories of paths from macro to hierarchical instance. *Data_type*: bool, optional

`-macro_to_inst_group`	Creates categories of paths from macro to instance group. *Data_type*: bool, optional
`-macro_to_stdcell`	Creates categories of paths from macro to standard cell. *Data_type*: bool, optional
`-macros` *macro_list*	Specifies the names of the macros to be used. Use this parameter with the `-partitions` parameter. *Data_type*: string, required
`-master` *master_category_name*	Specifies that the timing paths should be categorized only for the specified master category. *Data_type*: string, optional
`-partitions` *partition_list*	Specifies the names of the partitions to consider for extracting hierarchical information. *Data_type*: string, required
`-port_to_bbox`	Creates categories of paths from port to black box. *Data_type*: bool, optional
`-port_to_port`	Creates categories of paths from port to port. *Data_type*: bool, optional
`-port_to_macro`	Creates categories of paths from port to macro. *Data_type*: bool, optional
`-port_to_hinst`	Creates categories of paths from port to hierarchical instance. *Data_type*: bool, optional
`-port_to_inst_group`	Creates categories of paths from port to instance group. *Data_type*: bool, optional
`-port_to_stdcell`	Creates categories of paths from port to std cell. *Data_type*: bool, optional
`-stdcell_to_bbox`	Creates categories of paths from standard cell to black box. *Data_type*: bool, optional

`-stdcell_to_port`	Creates categories of paths from standard cell to port. *Data_type*: bool, optional
`-stdcell_to_macro`	Creates categories of paths from standard cell to macro. *Data_type*: bool, optional
`-stdcell_to_hinst`	Creates categories of paths from standard cell to hierarchical instance. *Data_type*: bool, optional
`-stdcell_to_inst_group`	Creates categories of paths from standard cell to instance group. *Data_type*: bool, optional
`-stdcell_to_stdcell`	Creates categories of paths from standard cell to standard cell. *Data_type*: bool, optional
`-types type_list`	Creates categories of paths within the type_list. Valid types are: `port`, `macro`, `bbox`, `hinst`, `stdcell` *Data_type*: string, required
`-use_current_floorplan`	Uses the current floorplan. *Data_type*: bool, required

Examples

- By default, it categorizes all paths inter/intra the 5 hierarchical entities by following command:
  ```
  analyze_paths_by_hierarchy -use_current_floorplan -all
  ```
- The following command categorizes all paths from "port" or to "port":
  ```
  analyze_paths_by_hierarchy -use_current_floorplan -type port
  ```

Related Topics

The Timing Budgeting chapter of the *Innovus User Guide* document.

analyze_paths_by_view

```
analyze_paths_by_view
[-help]
[-prefix string]
[-master master_category_name]
```

Categorizes the timing paths based on views that you create during multi-mode multi-corner analysis. By arranging the timing information of different views quickly, the command helps in analyzing timing paths.

For more information on creating path categories, see the 'Creating Path Categories' section in the Debugging Timing Results chapter of the *Innovus User Guide* document.

The command can be applied wherever the user loads the machine readable timing report.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `analyze_paths_by_view` parameter. For a detailed description of the command and all its parameters, use the `man` command: `man analyze_paths_by_view`
`-master` *`master_category_name`*	Specifies that the timing paths should be categorized only for the specified master category. *Data_type*: string, optional
`-prefix` *`string`*	Adds a prefix to the category name. *Data_type*: string, optional

Example

The following command categorizes paths based on views that you created during mmmc analysis:

```
analyze_paths_by_view
```

Related Topics

The Timing Budgeting chapter of the *Innovus User Guide* document.

create_histogram_png

```
create_histogram_png
[-help]
[-height integer]
-name string
[-width integer]
```

Dumps out the timing histogram image from global timing debug in the PNG format without bringing up the GUI. Earlier, this was done when the GUI was displayed by right-clicking the histogram window and selecting Snapshot. But that worked only when the snapshot was displayed in the GUI. This command has eliminated the necessity of bringing up the GUI to dump out the timing histogram.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `create_histogram_png` parameter. For a detailed description of the command and all its parameters, use the `man` command: `man create_histogram_png`
`-height integer`	Specifies the height of the PNG image. *Minimum:* `1` *Maximum:* `10000` *Data_type*: int, optional
`-name string`	Specifies the name of the PNG file. *Data_type*: string, required
`-width integer`	Specifies the width of the PNG image. *Minimum:* `1` *Maximum:* `10000` *Data_type*: int, optional

Related Topics

The Timing Budgeting chapter of the *Innovus User Guide* document.

create_path_category

```
create_path_category
[-help] [-capture_latency string]
[-check_type string]
[-clock string]
[-comment string]
[-critical_false_path string]
[-delay_of_any_inst string]
[-delay_of_any_inst_not_of_lib_cell string]
[-delay_of_any_inst_not_of_name string]
[-delay_of_any_inst_of_name string]
[-delay_of_any_inst_of_lib_cell string]
[-delay_of_any_net string]
[-delta_delay_of_every_net string]
[-delta_delay_of_any_net string]
[-delay_of_any_net_not_of_name string]
[-delay_of_any_net_of_name string]
[-delay_of_every_inst string]
[-delay_of_every_inst_of_lib_cell string]
[-delay_of_every_inst_not_of_lib_cell string]
[-delay_of_every_inst_not_of_name string]
[-delay_of_every_inst_of_name string]
[-delay_of_every_net string]
[-delay_of_every_net_not_of_name string]
[-delay_of_every_net_of_name string]
[-file string]
[-from_lib_cell string]
[-from_clock string]
[-from_clock_edge string]
[-from_inst string]
[-from_inst_pointer string]
[-from_pdomain string]
[-from_pin string]
[-from_port string]
[-launch_latency string]
[-master string]
[-name string]
[-not_from_cell string]
[-not_from_clock string]
[-not_from_clock_edge string]
[-not_from_inst string]
```

[-not_from_pdomain *string*]
[-not_from_pin *string*]
[-not_from_port *string*]
[-from_port_pointer *string*]
[-not_through_lib_cell *string*]
[-not_through_inst *string*]
[-not_through_net *string*]
[-not_through_pdomain *string*]
[-not_through_pin *string*]
[-not_through_port *string*]
[-not_to_lib_cell *string*]
[-not_to_clock *string*]
[-not_to_clock_edge *string*]
[-not_to_inst *string*]
[-not_to_pdomain *string*]
[-not_to_pin *string*]
[-not_to_port *string*]
[-number_of_insts *string*]
[-number_of_insts_not_of_name *string*]
[-number_of_insts_not_of_lib_cell *string*]
[-number_of_insts_of_lib_cell *string*]
[-number_of_insts_of_name *string*]
[-overwrite]
[-sdc *string*]
[-skew *string*]
[-slack *string*]
[-through_lib_cell *string*]
[-through_inst *string*]
[-through_net *string*]
[-through_pdomain *string*]
[-through_pin *string*]
[-through_port *string*]
[-to_lib_cell *string*]
[-to_clock *string*]
[-to_clock_edge *string*]
[-to_inst *string*]
[-to_inst_pointer *string*]
[-to_pdomain *string*]
[-to_pin *string*]
[-to_port *string*]
[-to_port_pointer *string*]
[-total_delay_of_insts *string*]
[-total_delay_of_insts_not_of_name *string*]

```
[-total_delay_of_insts_not_of_lib_cell string]
[-total_delay_of_insts_of_lib_cell string]
[-total_delay_of_insts_of_name string]
[-total_delay_of_level_of_chain_of_inst_type string]
[-total_delay_of_nets string]
[-total_delay_of_nets_not_of_name string]
[-total_delay_of_nets_of_name string]
[-total_delay_of_path string]
[-total_delta_delay_of_path string]
[-uncertainty string]
[-view string]
```

This command creates path categories using conditions for grouping paths according to analyzed timing results. The command can be applied wherever the user loads the machine readable timing report. For more information on analyzing timing results and, see 'Analyzing Timing Results' and 'Creating Path Categories' sections in the Debugging Timing Results chapter of the *Innovus User Guide* document.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `create_path_category` parameter. For a detailed description of the command and all its parameters, use the `man` command: `man create_path_category`
`-check_type {Setup Check | Hold Check | Clock Gating Setup Check | Library Clock Gating Setup Check | Clock Gating Hold Check | Library Clock Gating Hold Check | External Delay Assertion | Latch Borrowed Time Check | PulseWidth Check | ClockPeriod Check | Recovery Check | Removal Check}`	Indicates that the check type matches the specified option. *Data_type*: string, optional

`-clock {both | launch | capture}`	Helps you search the Clock Tree instead of the data path. If this option is not specified, the user searches the data path by default. You can use one of the following values: • `Both`: searches on launch or capture clock paths. • `Launch`: searches only on launch clock paths • `Capture`: searches only on capture clock paths **Note:** Supports simultaneous criteria on data path. *Data_type*: string, optional
`-comment {`*`commentText`*`}`	Specifies a comment that is associated with the category. *Data_type*: string, optional
`-critical_false_path`	*Data_type*: string, optional
`-master` *`existing_category_name`*	Specifies that the new category should be created as a nested category of the specified master category. The master category should be an existing category. *Default:* If you do not specify this parameter, the category is created as a new category at the top level. *Data_type*: string, optional
`-name` *`group_name`*	Specifies the name of the category to be created. *Data_type*: string, optional

`-overwrite`	Overwrites any predefined category with the same name in your category list. *Data_type*: bool, optional
`-sdc {`*`file_name line_number`*`}`	Creates a category based on the SDC constraint as per the specified SDC file name and line number. Creating a path category based on an SDC constraint helps you to analyze the affect of an SDC constraint by grouping the affected paths together. *Data_type*: string, optional
`-file` *`string`*	Specifies the file name. *Data_type*: string, optional
`-from_inst | -from_inst_pointer | -from_pin | -from_lib_cell | -from_clock | -from_port | -from_port_pointer` *`string`*	Indicates that the path must be from the specified instance, pin, library cell, clock, or port pattern. *Data_type*: string, optional
`-from_pdomain` *`string`*	Indicates that the path must be from the specified power domain. *Data_type*: string, optional
`-from_clock_edge`	Indicates that the path must be from the specified clock edge. *Data_type*: string, optional
`-not_from_inst | -not_from_pin | -not_from_cell | -not_from_clock | -not_from_port` *`string`*	Indicates that the path must not be from the specified instance, pin, cell, clock, or port pattern. *Data_type*: string, optional

`-not_from_pdomain` *string*	Indicates that the path must not be from the specified power domain. *Data_type*: string, optional
`-through_inst | -through_pin | -through_lib_cell | -through_net | -through_port` *string*	Indicates that the path must be through the specified instance, pin, net, library cell, or port pattern. *Data_type*: string, optional
`-through_pdomain` *string*	Indicates that the path must be through the specified power domain. *Data_type*: string, optional
`-not_to_clock_edge`	Indicates that the path must not be to the specified clock edge. *Data_type*: string, optional
`-slack` *float*	Indicates that the slack of the path must match the specified *float* value. *Data_type*: string, optional
`-not_from_clock_edge`	Indicates that the path must not be from the specified clock edge. *Data_type*: string, optional
`-not_through_inst | -not_through_pin | -not_through_net | -not_through_port | -not_through_pdomain` *string*	Indicates that the path must not be through the specified instance, pin, power domain, net, or port pattern. *Data_type*: string, optional
`-not_through_lib_cell` *string*	Indicates that the path must not be through the specified library cell. *Data_type*: string, optional

`-to_inst	-to_inst_pointer	-to_pin	-to_lib_cell	-to_clock	-to_port	-to_port_pointer	-to_pdomain` *string*	Indicates that the path must be the specified instance, pin, clock, power domain, library cell, or port pattern. *Data_type*: string, optional
`-not_to_inst	-not_to_pin	-not_to_lib_cell	-not_to_clock	-not_to_port	-not_to_pdomain` *string*	Indicates that the path must not be to the specified instance, pin, clock, library cell, power domain, or port pattern. *Data_type*: string, optional
`-skew` *float*	Indicates that the skew of the path must match the specified *float* value. *Data_type*: string, optional
`-launch_latency` *float*	Indicates that the launch latency of the path must match the specified *float* value. *Data_type*: string, optional
`-capture_latency` *float*	Indicates that the capture latency of the path must match the specified *float* value. *Data_type*: string, optional
`-uncertainty` *float*	Indicates that the uncertainty of the path must match the specified *float* value. *Data_type*: string, optional
`-number_of_insts` *integer*	Indicates that the number of instances in the data segment (with launch and capture clock paths) must match the specified *integer* value. *Data_type*: string, optional

`-number_of_insts_of_name` *`string_plus_integer`*	Indicates that the number of instances matching the specified instance name match the specified *`integer`* value. *Data_type*: string, optional
`-number_of_insts_not_of_name` *`string_plus_integer`*	Indicates that the number of instances not matching the specified instance name match the specified *`integer`* value. *Data_type*: string, optional
`-number_of_insts_of_lib_cell` *`string`*	Indicates that the number of instances matching the specified library cell match the specified *`integer`* value. *Data_type*: string, optional
`-number_of_insts_not_of_lib_cell` *`string`*	Indicates that the number of instances not matching the specified library cell match the specified *`integer`* value. *Data_type*: string, optional
`-to_clock_edge`	Indicates that the path must be to the specified clock edge. *Data_type*: string, optional
`-total_delay_of_insts` *`float`*	Indicates that the total delay of instances matches the specified *`float`* value. *Data_type*: string, optional
`-total_delay_of_level_of_chain_of_inst_type`	Indicates that the total delay of level of chain matching the specified insttype matches the specified string value. *Data_type*: string, optional

`-total_delay_of_path` *float*	Indicates that the total delay of path matches the specified *float* value. *Data_type*: string, optional
`-total_delta_delay_of_path` *float*	Indicates that the total incremental delay of path matches the specified *float* value. *Data_type*: string, optional
`-total_delay_of_nets` *float*	Indicates that the total delay of nets matches the specified *float* value. *Data_type*: string, optional
`-total_delay_of_insts_of_name` *string_plus_float*	Indicates that the total delay of instances matching the specified name matches the specified *float* value. *Data_type*: string, optional
`-total_delay_of_insts_not_of_name` *string_plus_float*	Indicates that the total delay of instances not matching the specified name matches the specified *float* value. *Data_type*: string, optional
`-total_delay_of_insts_not_of_lib_cell` *string*	Indicates that the total delay of instances not matching the specified library cell matches the specified *float* value. *Data_type*: string, optional
`-total_delay_of_insts_of_lib_cell` *string*	Indicates that the total delay of instances matching the specified library cell matches the specified *float* value. *Data_type*: string, optional

`-total_delay_of_nets_of_name` *`string_plus_float`*	Indicates that the total delay of nets matching the specified name matches the specified *`float`* value. *Data_type*: string, optional
`-total_delay_of_nets_not_of_name` *`string_plus_float`*	Indicates that the total delay of nets not matching the specified name matches the specified *`float`* value. *Data_type*: string, optional
`-delay_of_any_inst` *`float`*	Indicates that the delay of any instance in the design matches the specified *`float`* value. *Data_type*: string, optional
`-delay_of_any_inst_not_of_lib_cell` *`float`*	Indicates that the total delay of of any instance in the design not matching the specified library cell matches the specified *`float`* value. *Data_type*: string, optional
`-delay_of_any_net` *`float`*	Indicates that the delay of any net in the design matches the specified *`float`* value. *Data_type*: string, optional
`-delta_delay_of_any_net` *`float`*	Indicates that the incremental delay of any net in the design matches the specified *`float`* value. *Data_type*: string, optional
`-delta_delay_of_every_net` *`float`*	Indicates that the incremental delay of every net in the design matches the specified *`float`* value. *Data_type*: string, optional

`-delay_of_every_inst` *`float`*	Indicates that the delay of every instance in the specified library cell matches the specified *`float`* value. *Data_type*: string, optional
`-delay_of_every_inst_not_of_lib_cell` *`float`*	Indicates that the delay of every instance in the design not matching the specified library cell matches the specified *`float`* value. *Data_type*: string, optional
`-delay_of_every_net` *`float`*	Indicates that the delay of every instance in the design matches the specified *`float`* value. *Data_type*: string, optional
`-delay_of_any_inst_of_lib_cell` *`float`*	Indicates that the delay of any instance in the specified library cell matches the specified *`float`* value. *Data_type*: string, optional
`-delay_of_any_inst_of_name` *`string_plus_float`*	Indicates that the delay of any instance in the design matching the specified name matches the specified *`float`* value. *Data_type*: string, optional
`-delay_of_any_inst_not_of_name` *`string_plus_float`*	Indicates that the delay of any instance in the design not matching the specified name matches the specified *`float`* value. *Data_type*: string, optional

`-delay_of_any_net_of_name` *`string_plus_float`*	Indicates that the delay of any net in the design matching the specified name matches the specified *`float`* value. *Data_type*: string, optional
`-delay_of_any_net_not_of_name` *`string_plus_float`*	Indicates that the delay of any net in the design not matching the specified name matches the specified *`float`* value. *Data_type*: string, optional
`-delay_of_every_inst_of_name` *`string_plus_float`*	Indicates that the delay of every instance in the design matching the specified name matches the specified *`float`* value. *Data_type*: string, optional
`-delay_of_every_inst_of_lib_cell` *`string`*	Indicates that the delay of every instance in the design matching the specified name matches the specified *`float`* value. *Data_type*: string, optional
`-delay_of_every_inst_not_of_name` *`string_plus_float`*	Indicates that the delay of every instance in the design not matching the specified name matches the specified *`float`* value. *Data_type*: string, optional
`-delay_of_every_net_of_name` *`string_plus_float`*	Indicates that the delay of every net in the design matching the specified name matches the specified *`float`* value. *Data_type*: string, optional

`-delay_of_every_net_not_of_name` *`string_plus_float`*	Indicates that the delay of every net in the design not matching the specified name matches the specified *`float`* value. *Data_type*: string, optional
`-view`	Specifies the view name. *Data_type*: string, optional

Examples

- The following command categorizes paths whose source clock is `vclk1`:
  ```
  create_path_category -name vclk1 -from_clock vclk1
  ```
  The category `vclk1` is created.
- The following command categorizes paths whose skew is larger than 0.2:
  ```
  create_path_category -name worst_skew -skew {> 0.2}
  ```
  The category `worst_skew` is created.
- The following command creates paths from the macros in the design:
  ```
  create_path_category -name from_RAM_128x16 -from_inst RAM_128x16
  ```
- The following command creates paths to the macros in the design:
  ```
  create_path_category -name to_RAM_128x16 -to_inst RAM_128x16
  ```
- The following command creates paths between the macros in the design:
  ```
  create_path_category -name between_RAM_128_256 -from_inst RAM_128x16 -to_inst
  RAM_256x16
  ```

Related Topics

The Timing Budgeting chapter of the *Innovus User Guide* document.

delete_path_category

```
delete_path_category
[-help]
category_name
```

Deletes the selected category name.

The command can be applied when categories are created in Global Timing Debug.

For more information on creating path categories, see the 'Creating Path Categories' section in the Debugging Timing Results chapter of the *Innovus User Guide* document.

Parameter

`-help`	Outputs a brief description that includes the type and default information for each `delete_path_category` parameter. For a detailed description of the command and all its parameters, use the `man` command: `man delete_path_category`
`category_name`	Specifies the name of the category to be deleted. *Data_type*: string, optional

Example

The following command deletes the category `worst_paths`:

```
delete_path_category worst_paths
```

Related Topics

The Timing Budgeting chapter of the *Innovus User Guide* document.

gui_gtd_highlight_timing_report

```
gui_gtd_highlight_timing_report
[-help]
[-append]
[-color_index integer]
[-in_file filename]
[-no_arrow]
[-paths path_number | -all]
```

Highlights the selected paths in the user interface.

For more information on creating path categories, see the 'Creating Path Categories' section in the Debugging Timing Results chapter of the *Innovus User Guide* document.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `gui_gtd_highlight_timing_report` parameter. For a detailed description of the command and all its parameters, use the `man` command: `man gui_gtd_ighlight_timing_report`
`-all`	Highlights all the paths. The option is of type boolean and required. *Data_type*: bool, required
`-append`	Appends a path to the current highlight. *Data_type*: bool, required
`-color_index` *integer*	Specifies the `color_index` of the path from 1 to 8. The option is of type int and optional with minimum value 1, and maximum value 8. *Data_type*: int, optional
`-in_file` *filename*	Name of the file. *Data_type*: string, optional
`-no_arrow`	Removes the highlighted arrow. *Data_type*: bool, optional
`-paths` *path_number*	Specify the number of the path. *Data_type*: string, required

Related Topics

The Timing Budgeting chapter of the *Innovus User Guide* document.

gui_open_gtd

```
gui_open_gtd
[-help]
```

Opens the global timing debugger in the common user interface mode for accessing the GTD files report.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `gui_open_gtd` parameter. For a detailed description of the command and all its parameters, use the `man` command: `man gui_open_gtd`

read_path_categories

```
read_path_categories
[-help]
in_file filename
```

Loads the path category file in the Timing Debug form. The path category file contains category definitions that you saved in an earlier timing debug analysis. To save path categories, use the `write_path_categories` command.

For more information on creating path categories, see the 'Creating Path Categories' section in the Debugging Timing Results chapter of the *Innovus User Guide* document.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `read_path_categories` parameter. For a detailed description of the command and all its parameters, use the `man` command: `man read_path_categories`
`in_file` *`filename`*	Specifies the name of the category file. *Data_type*: string, required

Example

The following command loads the path category file `category.cat`:

```
read_path_categories in_file category.cat
```

Related Topics

- `write_path_categories`
- The Timing Budgeting chapter of the *Innovus User Guide* document.

read_timing_debug_report

```
read_timing_debug_report
[-help]
[-name report_name]
[{-proto [-slack_margin number]  [-num_path number]}]
[-num_path]
timing_report_filename
```

Loads the violation report in Innovus for debugging timing results.

Note: The violation report that you specify in this command has a special machine readable format. Use the `report_timing -machine_readable` command or `time_design -timing_debug_report` command to generate the machine readable violation report.

For more information on creating path categories, see the 'Creating Path Categories' section in the Debugging Timing Results chapter of the *Innovus User Guide* document.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `read_timing_debug_report` parameter. For a detailed description of the command and all its parameters, use the `man` command: `man read_timing_debug_report`
`timing_report_filename`	Specifies the name of the timing debug timing_report_file. *Data_type*: string, required
`-name`	Specifies the report name to load the machine readable timing report timing_report_files incrementally. *Data_type*: string, optional
-num_path	Specifies the number of paths to be reported in the detailed path violations. *Data_type*: int, optional
-proto	Creates flex model categories and displays the top path of the top 8 (by default) categories. This option can be used for design that has flexModels. *Data_type*: bool, required
-slack_margin number	Specifies the additional specified slack. *Data_type*: float, optional

Examples

- The following command loads the machine readable timing report `top.mtarpt`:
  ```
  read_timing_debug_report top.mtarpt
  ```
- The following commands load two machine readable timing reports `top1.mtarpt` and `top2.mtarpt` together:
  ```
  read_timing_debug_report -name default_report top1.mtarpt
  read_timing_debug_report -name default_report top2.mtarpt
  ```
  GTD will first load the report `top1.mtarpt`, then append `top2.mtarpt` incrementally to the

current report.
Note: The `name` can be specified by user with any string, it is not the `timing_report_file` name.

Related Topics

The Timing Budgeting chapter of the *Innovus User Guide* document.

write_category_summary

```
write_category_summary
[-help]
out_file <string>
[[-sort_low_to_high | -sort_high_to_low ]
[-sort_by_tns | -sort_by_wns ]]
[-csv | {-no_frame_fix_width [-no_split ]}]
```

Writes a text file containing the following information.

- Category name
- Total number of paths
- Number of passing paths
- Number of failing paths
- Worst negative slack
- Total negative slack

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `write_category_summary` parameter. For a detailed description of the command and all its parameters, use the `man` command: `man write_category_summary`
`-sort_low_to_high`	Specifies that the report in ascending order, should specify option `-sort_by_tns` or `-sort_by_wns`. *Data_type*: bool, optional
`-sort_by_tns`	The TNS value specified by the sorting key through the option `-sort_low_to_high` or `-sort_high_to_low`. *Data_type*: bool, optional

`-sort_by_wns`	The WNS value specified by the sorting key through the option `-sort_low_to_high` or `-sort_high_to_low`. *Data_type*: bool, optional
`-csv`	Reports entries in csv format. *Data_type*: bool, optional
`-sort_high_to_low`	Specifies that the report in descending order, should specify option `-sort_by_tns` or `-sort_by_wns`. *Data_type*: bool, optional
`-no_frame_fix_width`	Specifies that the table columns have a fixed minimum width, but no maximum width (the data does not wrap). Column data that fits within the column's minimum width will left-justify and pad its location to fill the specified column space. Column data that is longer than the column's minimum width will take as much space in the table as needed. If a column is overwritten by the data from the preceding column, its data will begin on a new line at the correct column location. *Default:* `-frame` *Data_type*: bool, optional
`-no_split`	Specifies that each column follows the preceding column without splitting to a new line, if the data in one column overruns the next column. The wrapping behavior for data is controlled by the size of the display window. This format is useful for machine parsing the report. **Note:** You must specify `-no_frame_fix_width` in order to use this parameter. *Data_type*: bool, optional
out_file `fileName`	Specifies the name of the category report file. *Default*: `category.rpt` *Data_type*: string, required

Examples

The following command writes a category summary file `category.rpt`:

```
write_category_summary out_file category.rpt
```

Related Topics

The Timing Budgeting chapter of the *Innovus User Guide* document.

write_histogram_view

```
write_histogram_view
[-help]
file_name string
[-type string]
```

Saves (dumps) the image of the histogram shown in *Global Timing Debug* automatically during the process.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `write_histogram_view` parameter. For a detailed description of the command and all its parameters, use the `man` command: `man write_histogram_view`
`file_name string`	Specifies the name of the image file to save the histogram. *Data_type*: string, required
`-type string`	Specifies the type of file (image format) to be saved. Options are `bmp/jpg/gif/tif/csv`. *Default*: `bmp` *Data_type*: string, optional

Related Topics

The Timing Budgeting chapter of the *Innovus User Guide* document.

write_path_categories

```
write_path_categories
[-help]
out_file filename
```

Saves the selected categories in a path category file. You can use the category file to load saved category definitions in a new timing debug session. To load the saved category definitions, use the `read_path_categoriess` command.

For more information on creating path categories, see the 'Creating Path Categories' section in the Debugging Timing Results chapter of the *Innovus User Guide* document.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `write_path_categories` parameter. For a detailed description of the command and all its parameters, use the `man` command: `man write_path_categories`
`out_file` *`filename`*	Specifies the name of the category file. *Data_type*: string, required

Example

The following command saves the path category file `category.cat`:

```
write_path_categories out_file category.cat
```

Related Topics

- `read_path_categories`
- The Timing Budgeting chapter of the *Innovus User Guide* document.

write_path_list_summary

```
  write_path_list_summary
[-help]
[-category string]
[-compress]
[-csv]
-report fileName
```

Writes the Path List categorized summary into a text and/or CSV file.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `write_path_list_summary` parameter. For a detailed description of the command and all its parameters, use the `man` command: `man write_path_list_summary`
`-category string`	Specifies the category of paths to be stored into the CSV file. *Data_type*: string, optional
`-compress`	Compresses the output file in the `.gzip` file format. *Data_type*: bool, optional
`-csv`	Reports entries in the CSV format. *Data_type*: bool, optional
`-report fileName`	Specifies the name of the CSV output file. *Data_type*: string, required

Related Topics

The Timing Budgeting chapter of the *Innovus User Guide* document.

write_text_timing_report

```
write_text_timing_report
[-help]
[-machine_readable file.mtarpt |
-category category_name]
[-format{{column1 width1} {column2 width2} ...} ]
[-path_spec path_specification]
[-summary] |
report_file file.tarpt
```

Writes a text file containing information about paths that have been loaded from a `.mtarpt` file from Timing Debug, either with the `load_timing_debug_report` command or in the *Display/Generate Timing Report* form in the GUI.

Paths are selected as follows:

- If only one `.mtarpt` file is loaded and no other parameters are specified, then the tool writes all the paths contained in the loaded `.mtarpt` file.
- If there are several `.mtarpt` files loaded, you must specify the `.mtarpt` file from which the tool can select the paths, except if a category is specified.

You can use this report for performing timing analysis.

If only one machine-readable report file is loaded, then the tool writes all paths from that file into the output report. If more than one file is loaded, then the tools issues an error message.

Note: The generated report cannot be read back into the timing debug environment.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `write_text_timing_report` parameter. For a detailed description of the command and all its parameters, use the `man` command: `man write_text_timing_report`
`-category category_name`	

	Specifies the category of paths to write to the text file. You cannot specify `-machine_readable` if you specify this parameter. *Data_type*: string, optional
`-format{{`*`column1 width1`*`} {`*`column2 width2`*`} ...}`	
	Formats the report according to the specified column name and its width. The column-width pair specifies the column to display in the report and the width of that column. Multiple column names can be specified. *Data_type*: string, optional
`-machine_readable` *`file.mtarpt`*	
	Specifies the name of the `.mtarpt` file to read and convert to a text file. You cannot specify this parameter if you specify `-category`. *Data_type*: string, optional
`-path_spec` *`path_specification`*	
	Writes a single number or path range separated by blank space. A path range is two numbers separated by a minus "-" sign. You can use this parameter only if a single file is specified or loaded. *Data_type*: string, optional
`report_file` *`file.tarpt`*	
	Specifies the name of the text output file. *Data_type*: string, required
`-summary`	Writes the file header only. Use this parameter if you do not need a detailed table. *Data_type*: bool, optional

Examples

- The following command loads machine-readable report `top.mtarpt`:
  `load_timing_debug_report top.mtarpt`
- The following command reads the `top.mtarpt` report and reports paths 1 through 50 in the `top_tarpt.txt` file.

```
write_text_timing_report -machine_readable top.mtarpt -path_spec {1-50}
report_file top_tarpt.txt
```

Related Topics

The Timing Budgeting chapter of the *Innovus User Guide* document.

41

Timing Modeling Commands

- check_timing_model
- merge_model_timing
- report_timing_model
- write_timing_model

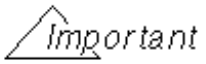
check_timing_model

```
check_timing_model
[-help]
[-absolute_cap_tolerance float]
[-absolute_tolerance float]
[-absolute_transition_tolerance float]
-compare_file string
[-ignore_view]
-out_file string
[-percentage_cap_tolerance float]
[-percentage_tolerance float]
[-percentage_transition_tolerance float]
-reference_file string
```

Compares two reports that contain interface timing characteristics generated by the `report_timing_model` command.

This command compares the interface timing report generated by using the `report_timing_model` command on the original netlist with the interface timing report generated on the model extracted for the same design.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `check_timing_model` parameter. For a detailed description of the command and all of its parameters, use the man command: `man check_timing_model`.
`-absolute_cap_tolerance` *float*	Specifies the tolerance value for capacitance (as an absolute value) while comparing two report files. This parameter specifies the permissible difference, as an absolute value, between the capacitance values of two report files. If the difference in the value of capacitance exceeds the absolute tolerance value specified, the comparison status is set to `FAIL`. **Note:** When this parameter is not specified, the software uses the absolute value specified with the `-absolute_tolerance` parameter. If the `-absolute_tolerance` parameter is not specified, the value defaults to `0`.  When the `-percentage_cap_tolerance` and `-absolute_cap_tolerance` parameters are used together, the comparison status is set to `FAIL` only when the difference in the value of capacitance exceeds the specified absolute as well as percentage tolerance values.
`-absolute_tolerance` *float*	Specifies the absolute tolerance value. The tolerance value is used when there is a difference between the timing characteristics of the reference file (`-reference_file`) and the file that is being compared (`-compare_file`) to determine the `PASS` or `FAIL` status of the comparison. If the difference in the value of a timing characteristic exceeds the absolute tolerance value, the comparison status is set to `FAIL`. **Note:** When used with the `-percentage_tolerance` option, the difference in the value of a timing characteristic should exceed both absolute and percentage tolerance values for the comparison result status of an arc to be `FAIL`. *Default*: `0`

`-absolute_transition_tolerance` *`float`*	Specifies the name of the report file that contains the interface timing characteristics extracted from the model. The timing characteristics of this file are compared with the timing characteristics of the reference file that you specify with the `-reference_file` option.
`-compare_file` *`string`*	Specifies the name of the report file that contains the interface timing characteristics extracted from the model created using the model extraction command.
`-ignore_view`	Compares non-MMMC design `report_timing_model` output report with the MMMC design output report (of `report_timing_model` command). When you specify this parameter, the software ignores the view information while comparing the reports. So you do not require to create dummy views for timing model validation flow.
`-out_file` *`string`*	Specifies the name of the output file in which the comparison results will be generated. This file contains the `PASS` or `FAIL` status for the timing data comparison.

`-percentage_cap_tolerance` *float*	Specifies the tolerance value for capacitance (as a percentage) while comparing two report files. This parameter specifies the permissible difference, in percentage, between the capacitance values of two report files. If the percentage difference in the value of capacitance exceeds the percentage tolerance value specified, the comparison status is set to `FAIL`. **Note:** When this parameter is not specified, the software uses the percentage value specified with the `-percentage_tolerance` parameter. If the `-percentage_tolerance` parameter is not specified, the value defaults to `0`. When the `-percentage_cap_tolerance` and `-absolute_cap_tolerance` parameters are used together, the comparison status is set to `FAIL` only when the difference in the value of capacitance exceeds the specified absolute as well as percentage tolerance values.
`-percentage_tolerance` *float*	Specifies the percentage tolerance value for comparing the timing data of two report files. The percentage tolerance value is used when there is a difference between the timing characteristics of the reference file (`-reference_file`) and the file that is being compared (`-compare_file`) to determine the `PASS` or `FAIL` status of the comparison. If the percentage difference in the value of a timing characteristic exceeds the percentage tolerance value, the comparison status is set to `FAIL`. **Note:** When used with the `-absolute_tolerance` option, the difference in the value of a timing characteristic should exceed both percentage and absolute tolerance values for the comparison result status of an arc to be `FAIL`. *Default*: `0`

`-percentage_transition_tolerance` *float*	Specifies the tolerance value for transition time at ports (as a percentage) while comparing two report files.This parameter specifies the permissible difference, in percentage, between the transition time at ports of two report files. If the percentage difference in the transition time exceeds the percentage tolerance value specified, the comparison status is set to `FAIL`. **Note:** When this parameter is not specified, the software uses the percentage value specified with the `-percentage_tolerance` parameter. If the `-percentage_tolerance` parameter is not specified, the value defaults to `0`. When the `-percentage_transition_tolerance` and `-absolute_transition_tolerance` parameters are used together, the comparison status is set to `FAIL` only when the difference in the value of transition exceeds the specified absolute as well as percentage tolerance values.
`-reference_file` *string*	Specifies the name of the reference file that is used for comparison. This is the report file that was generated by using the `write_timing_model` command on the original design. The file specified with the `-compare_file` option is compared with this file.

Examples

- The following command compares the timing characteristics of the `sample_compare.rpt` file with the `sample_reference.rpt` file and generates the comparison results in the `sample_output.rpt` file:

```
check_timing_model -reference_file sample_reference.rpt -compare_file
sample_compare.rpt
-out_file sample_output.rpt
```

```
#################################################################
# check_timing_model report
# Design : netlist = topcell =
# Version : 07.10-dev.indfe_cm
# Analysis Mode : setup
# Date : Wed Mar 14 15:56:59 IST 2007
#################################################################
#Section 1: Worst-case Arc-Delay
#Tolerance Percent: 0.0%
#Tolerance Absolute: 0.0
#NOTE: The various columns list values in the form 'X[Y]', where:
# X = value that is compared.
# Y = reference value.

#NOTE2: If a comparison fails (i.e. status = FAIL), then the clauses
# (i.e. 'X[Y]' pairs) that caused the failure are identified with
# a '*' in front of them.
#From   To    Arc-Type   Transition   Arc-Delay     abdDiff   %Diff  Status
#################################################################
in     CLK3   setup      fall/rise   0.134[0.134]   0.000   0.000%  PASS
in     CLK3   setup      rise/rise  -0.005[-0.005]  0.000  -0.000%  PASS
#################################################################
#Section 2: Pin-capacitances
#Tolerance: 0.0%
#NOTE: The various columns list values in the form 'X[Y]', where:
# X = value that is compared.
# Y = reference value.

#NOTE2: If a comparison fails (i.e. status = FAIL), then the clauses
# (i.e. 'X[Y]' pairs) that caused the failure are identified with
# a '*' in front of them.
#
#Pin C(total)[rise]  C(total)[fall] Status
#################################################################
```

```
clk   0.002[0.002]  0.002[0.002]    PASS
in    0.002[0.002]  0.002[0.002]    PASS
out   0.000[0.000]  0.000[0.000]    PASS
#################################################################
#Section 3: Transition-times
#Tolerance: 0.0%
#NOTE: The various columns list values in the form 'X[Y]', where:
# X = value that is compared.
# Y = reference value.
#
#NOTE2: If a comparison fails (i.e. status = FAIL), then the clauses
# (i.e. 'X[Y]' pairs) that caused the failure are identified with
# a '*' in front of them.
#
#Pin   Rise        Fall         Status
#################################################################
clk   0.000[0.000] 0.000[0.000] PASS
in    0.000[0.000] 0.000[0.000] PASS
out   0.056[0.056] 0.051[0.051] PASS

#################################################################
#Section 4: Design Rules
#Tolerance: 0.0%
#NOTE: The various columns list values in the form 'X[Y]', where:
# X = value that is compared.
# Y = reference value.

#NOTE2: If a comparison fails (i.e. status = FAIL), then the clauses
# (i.e. 'X[Y]' pairs) that caused the failure are identified with
# a '*' in front of them.
#
#Pin    MaxCap    MinCap    MaxTrans    MaxFanout    FanoutLoad    Status
#########################################################################
clk     NA[NA]     NA[NA]    NA[NA]      NA[NA]       NA[NA]        PASS
```

```
in      NA[NA]     NA[NA]    NA[NA]      NA[NA]       NA[NA]         PASS
out     NA[NA]     NA[NA]    NA[NA]      NA[NA]       NA[NA]         PASS

#########################################################################
# SUMMARY
#
# Check-Type Passed Failed N/A Total
#########################################################################
Worst-Slacks 2 0 0 2
Capacitances 3 0 0 3
Trans-Times 3 0 0 3
Design-Rules 3 0 0 3
```

Related Information

- report_timing_model
- write_timing_model

merge_model_timing

```
merge_model_timing
[-help]
-input_library_file string
[-keep_separate_arc]
[-merged_library_file string]
-mode_group string
-modes string
[-pin_cap_tolerance float]
[-tolerance float]
```

Merges extracted timing models (ETM) for various analysis views into a single output.

The `merge_model_timing` command should be the last command in the run script. Also, this command should be run separately in a session when a design is not loaded.

When the DRV values are different in the libraries that are merged, then the software saves the worst value of the DRV with a warning message.

If the DRV values are not equal, the greater value is stored for the following:

- Minimum transition
- Minimum capacitance
- Minimum fanout

The smaller value is stored for the following, in case of unequal DRVs:

- Maximum transition
- Maximum capacitance
- Maximum fanout
- Fanout load

Parameters

`-keep_separate_arc`	Enables creating separate timing arcs with respective modes.
`-input_library_file` *string*	Specifies input library files that are to be merged.
`-modes` *string*	Specifies the mode names required for merging the libraries. The number of mode names should be equal to the number of input libraries.
`-mode_group` *string*	Specifies the name of the mode group.
`-merged_library_file` *string*	Specifies the name of the merged library output file name. By default, the output file name is `merge.lib`.
`-pin_cap_tolerance` *float*	Specifies the pin capacitance tolerance for making tolerance comparison for library merging.
`-tolerance` *float*	Specifies the tolerance value for making tolerance comparison for library merging.

Examples

- The following command merges library files `model1.view1.lib` and `model1.view1.lib.ref` into a single output file:

```
merge_model_timing -input_library_file "model1.view1.lib model1.view1.lib.ref"
-modes "v1 v2" -mode_group v1_v2
```

Related Information

- `write_timing_model`

report_timing_model

```
report_timing_model
[-help]
model_timing_report_file
[-in_ports <string>]
[-nworst <integer>]
[-out_ports <string>]
[-slew_propagation {worst | path_based}]
[-type {arc | slack}]
[-verbose]
[-view <string>]
-setup string | -hold string
```

Writes the interface timing characteristics of the design in the specified timing model report file.

The report file contains the following sections:

- `Worst-Case Arc/Slack`: Contains details about the extracted arcs and the slack or delay across them depending on the argument specified with the `-type` option.
- `Transition Time`: Contains information about transition time at ports.
- `Capacitance`: Contains the capacitance values for the ports.
- `Design Rules`: Contains the design rules applied to the ports.

The command reports unconstrained arcs delays by default.

Parameters

`-help`	Prints out a brief description that includes type and default information for each `report_timing_model` parameter. For a detailed description of the command and all of its parameters, use the man command: man `report_timing_model`.
`-hold` *string*	Write the paths corresponding to early analysis.

`-in_ports {`*`port_list`*`}`	Restricts the `report_timing_model` report to the input ports specified with this parameter. As a result, the software considers the timing paths related to the specified input ports only, while generating the report. If this parameter is not specified, all the input ports in the design are considered while generating the report.
`model_timing_report_file`	Specifies the name of the interface timing model report file that will be generated with the interface timing characteristics of the loaded design.
`-nworst` *`value`*	Specifies the number of paths to be enumerated for each end point. **Note:** The extracted model will contain greater number of setup/hold paths per end point because it will have lesser end points compared to the original netlist. Therefore, it is recommended that you use a greater `-nworst` value such as`40` while writing a timing report for the extracted model to derive better correlation.
`-out_ports {`*`port_list`*`}`	Restricts the `report_timing_model` report to the output ports specified with this parameter. As a result, the software considers the timing paths related to the specified output ports only, while generating the report. If this parameter is not specified, all the output ports in the design are considered while generating the report.
`-setup` *`string`*	Writes the paths corresponding to late analysis.
`-slew_propagation {worst | path_based}`	Sets the slew propagation mode to use for dumping the timing characteristics of the design.
`-type {arc | slack}`	Determines whether to report arc delay values (`arc`) or the worst-slack values (`slack`).
`-view` *`view_name`*	Specifies a view name. The report can be generated for a specific view name.
`-verbose`	Prints the command progress details.

Example

- The following command extracts the interface timing characteristics of the currently loaded design and writes out the report file named `sample.rpt` with the arc delay information:

```
report_timing_model -type arc sample.rpt
```

```
Output:

#################################################################
# report_timing_model report
# Design : netlist = .//case1.v topcell = set_propagated_clock_tc2
# Version : 07.10-dev.indfe_cm
# Analysis Mode : setup
# Date : Wed Mar 14 15:29:57 IST 2007
# Command : report_timing_model -type arc sample.rpt
#################################################################
#################################################################
# Section 1 : Worst Case Arc
# Analysis Mode : setup
# Description : worst case arcs
#
# From To Arc Type Transition Arc-Delays
#####################################################
in CLK3 setup fall/rise 0.134
in CLK3 setup rise/rise -0.005
#####################################################
# Section 2 : Transition time
# Analysis Mode : setup
# Description : Actual transition times on the ports
#
# Port RiseTran FallTran
#####################################################
in 0.000 0.000
clk 0.000 0.000
out 0.056 0.051
#####################################################
# Section 3 : Capacitance
# Analysis Mode : setup
# Description : Total Cap on the ports
#
# Port CTotal(Rise) CTotal(Fall)
#######################################
in 0.002 0.002
```

```
clk 0.002 0.002
out 0.000 0.000
#########################################
# Section 4 : Design Rules
# Analysis Mode : setup
# Description : Design Rules on the ports
#
# Port MaxCap MinCap MaxTrans MaxFanout Fanout Limit
#################################################
in NA NA NA NA NA
clk NA NA NA NA NA
out NA NA NA NA NA
```

- In MMMC mode, the `report_timing_model` command displays the following report output:

```
#############################################
# report_timing_model report
# Design : netlist =
./../../../tc/tmps/testcases/tmps_ilm/DATA/mmmc/dtmf_chip/dtmf_chip.Route.opt.ILM.e
nc.dat/dtmf_chip.v topcell = dtmf_chip

# Analysis Mode : setup
# Date : Mon Sep 21 22:07:40 PDT 2009
# Command : report_timing_model -type arc sample.rpt
#############################################################################
#############################################################################
# Section 1 : Worst Case Arcs
# Analysis Mode : setup
# Description : worst case arcs
#
# From   To           Arc Type      Transition   Arc-Delays    View
#############################################################################
vclk2    tdigit[0]    max_sequential rise/fall   2.676   mission+worst-rcMax
vclk2    tdigit[0]    max_sequential rise/rise   2.831   mission+worst-rcMax
vclk2    tdigit[1]    max_sequential rise/fall   2.413   mission+worst-rcMax
vclk2    tdigit_flag  max_sequential rise/rise   2.58    mission+worst-rcMax
vclk2    tdigit[0]    max_sequential rise/fall   2.676   test+worst-rcMax
```

```
vclk2    tdigit[0]    max_sequential rise/rise   2.831   test+worst-rcMax
vclk2    tdigit[1]    max_sequential rise/fall   2.413   test+worst-rcMax
#####################################################
# Section 2 : Transition time
# Analysis Mode : setup
# Description : Actual transition times on the ports
# View : default
#
# Port                   RiseTran   FallTran
#####################################################
reset                     0.120     0.120
int                       0.120     0.120
port_pad_data_in[15]      0.120     0.120
port_pad_data_in[0]       0.120     0.120
scan_en                   0.120     0.120
scan_clk                  0.120     0.120
scan_in_1                 0.120     0.120
scan_in_2                 0.120     0.120
test_mode                 0.120     0.120
spi_data                  0.120     0.120
spi_fs                    0.120     0.120
refclk                    0.120     0.120
vcom                      1.025     1.138
#####################################################
# Section 3 : Capacitance
# Analysis Mode : setup
# Description : Total Cap on the ports
# View : default
#
# Port               CTotal(Rise) CTotal(Fall)
#####################################################
reset                   3.078     3.078
port_pad_data_in[0]     3.078     3.078
scan_en                 3.078     3.078
```

```
scan_clk                3.078     3.078
scan_in_1               3.078     3.078
vcop                    0.007     0.007
#################################################################################
# Section 4 : Design Rules
# Analysis Mode : setup
# Description : Design Rules on the ports
# View : default
#
# Port                MaxCap  MinCap  MaxTrans  MaxFanout  Fanout  Limit
#################################################################################
reset                   NA    NA      2.700     15.000     NA
int                     NA    NA      2.700     15.000     NA
port_pad_data_in[15]    NA    NA      2.700     15.000     NA
port_pad_data_in[14]    NA    NA      2.700     15.000     NA
port_pad_data_in[13]    NA    NA      2.700     15.000     NA
port_pad_data_in[12]    NA    NA      2.700     15.000     NA
```

Related Information

- check_timing_model
- write_timing_model

write_timing_model

```
write_timing_model
[-help]
model_file
[-absolute_tolerance_validation float]
[-absolute_tolerance_optimistic_validation float]
[-absolute_tolerance_pessimistic_validation float]
[-assertions constraint filename]
[-cell_name cell_name]
[-clock_transitions {clk_slew1 clk_slew2...}]
[-gain integer]
[-include_aocv_weights]
[-include_power_ground]
[-input_transitions {input_slew1 input_slew2...}]
[-lib_name lib_name]
[-max_num_loads value]
[-max_num_transitions value]
[-output_loads {output_load1 output_load2...}]
[-percent_tolerance_validation float]
[-percent_tolerance_optimistic_validation float]
[-percent_tolerance_pessimistic_validation float]
[-precision integer]
[-resolution float]
[-tolerance float]
[-validation_dir string]
[-verilog_shell_file filename]
[-verilog_shell_module top_module_name]
[-view string]
```

Builds a Liberty (`.lib`) format model for the top cell, which is the timing model equivalent of the original design. The model is extracted for the given analysis modes, PVT conditions, and parasitics. If any one of these parameters change, then you must extract the model again. The model is extracted for a range of input slews and load capacitances. If you do not specify these ranges, the software automatically computes the default values. The extracted model defines all of the top-level ports of the design as pins. Additionally, all other design pins that are targets of `create_clock` or `create_generated_clock` are retained as internal pins.

For model extraction, your design must be properly constrained. For example, no clock or data conflicts can be present. Use the `check_timing` command and correct all errors prior to model extraction.

Parameters

`-help`	Prints out the command usage.
`-absolute_tolerance_validation` *float*	Specifies the absolute tolerance value to be used during the validation of extracted ETM. For more details refer to `check_timing_model -absolute_tolerance`. The integer value specified with this parameter is read as ns/ps depending upon the time unit. *Default:* 3ps
`-absolute_tolerance_optimistic_validation` *float*	Specifies the absolute tolerance limit for flagging an optimistic failure. An optimistic path violating the absolute as well as percentage tolerance limit will be flagged as FAIL in the `check_timing_model` command output report. You can use this parameter when the `-validation_dir` is used with model extraction. *Default*: 15ps
`-absolute_tolerance_pessimistic_validation` *float*	Specifies the absolute tolerance limit for flagging a pessimistic failure. A pessimistic path violating the absolute as well as percentage tolerance limit will be flagged as FAIL in the `check_timing_model` command output report. You can use this parameter when the `-validation_dir` is used with model extraction. *Default*: 15ps

`-assertions` *`constraint_filename`*	Dumps the false and multicycle path exceptions associated with the internal pins, which will be preserved in the grey box models. The exceptions are dumped in SDC format and can be used while instantiating grey box models. *Default*: `model.asrt`
`-cell_name` *`cell_name`*	Specifies the cell name used in the timing model. *Default:* The name of the top cell to be extracted.
`-clock_transitions {`*`clk_slew1 clk_slew2...`*`}`	Specifies a range of slew values used at all clock pins. The list of values must be enclosed by curly braces. If this parameter is not specified, the software automatically computes the appropriate values.
`-gain` *`integer`*	Represents an increase in the number of delay arcs if a pin is removed. This option reduces the extracted model size in scenarios where removing a pin can result in increased model size. The gain at an internal pin can be defined as: `(number of incoming delay arcs * number of outgoing delay arcs) - (number of incoming delay arcs + number of outgoing delay arcs` If an internal pin has a gain greater than the specified number, then that pin needs to be preserved in order to reduce the extracted model size. *Default*: `0`

`-include_aocv_weights`	Writes the stage weight on individual arcs. Before using this option, it is mandatory that the AOCV libraries are read in and AOCV analysis is enabled. The weights written in ETM are not meant to derate the model. Model itself will be containing the derated delays. The stage weight extracted on arcs will be used to calculate the stage count of chip level paths going across the ETM. The stage weights in the extracted model are dumped as user defined attributes. This will require defining two attributes at arc level.
	• `define ("aocv_weight","timing","float");`
	• `define ("clock_aocv_weight","timing","float");`
	The first attribute will be used for combinational and trigger arc and will be dumped at arc level as below.
	`timing () { timing_type : combinational; timing_sense : positive_unate; aocv_weight : 4; . }`
	The check arcs will be dumped with two separate stage weights for data and clock.
	`timing () { timing_type : setup_rising; aocv_weight : 4; clock_aocv_weight : 3; . }`

`-input_transitions {`*`input_slew1 input_slew2 ...`*`}`	Specifies a range of slew values used at all input pins. Enclose the list of values with curly braces. If this parameter is not specified, the software automatically computes the appropriate values.
`-lib_name` *`lib_name`*	Specifies the library name used in the timing model. Write extracted timing models in the liberty format by specifying the `.lib` file suffix. *Default:* The name of the top cell to be extracted.
`-max_num_loads` *`value`*	Specifies the maximum number of output load values used for delay arc characterization. A low value of 1 or 2 improves runtime and memory usage but may significantly degrade model accuracy. Use this parameter only if the model size or model extraction time is critical. *Default:* `64`
`-max_num_transitions` *`value`*	Specifies the maximum number of input slews or clock slews used for delay and check arc characterization. Specifying a low value of 1 or 2 improves runtime and memory but may significantly degrade model accuracy. Using the `-max_num_transitions` parameter has a larger impact on runtime and memory than the `-max_num_loads` parameter. Use this parameter only if the model size or the model extraction time is critical. *Default:* `64` The `max_num_slews` and `max_num_loads` parameters specify only the maximum limits on how many slew and load points are chosen. The software scans the gates connected to the input and output ports to calculate the number of points.

model_file	Specifies the name of the output file. Avoid problems with rebind by using the -lib_name and -cell_name parameters instead of using the default names.
-output_loads {*output_load1output_load2 ...*}	Specifies a range of load values used at all output pins. Enclose the list of values by curly braces. If this parameter is not specified, the software automatically computes the appropriate values.
-include_power_ground	Allows power or ground related pin information to be extracted inside the extracted model.
-percent_tolerance_validation *float*	Specifies the percentage tolerance value to be used during the validation of extracted ETM. For more details refer to check_timing_model -percent_tolerance. *Default:* 3%
-percent_tolerance_optimistic_validation *float*	Specifies the percentage tolerance limit for flagging an optimistic failure. An optimistic path violating percentage as well as absolute tolerance limit will be flagged as FAIL in the check_timing_model command output report. You can use this parameter when the -validation_dir is used with model extraction. *Default*: 3%
-percent_tolerance_pessimistic_validation *float*	Specifies the percentage tolerance limit for flagging a pessimistic failure. A pessimistic path violating percentage as well as absolute tolerance limit will be flagged as FAIL in the check_timing_model command output report. You can use this parameter when the -validation_dir is used with model extraction. *Default*: 3%

`-post_check`	Checks the generated ETM library using Valus.
`-precision` *value*	Specifies the floating point precision used for the timing model. For example, the floating point precision for 3.0089 can be changed as follows:
	Precision Data Value in .lib 4 3.0089 3 3.009 2 3.01 1 3.0 *Default:* `4`
`-resolution` *value*	Specifies the magnitude of delay difference that can be ignored by the extractor. *Default:* `0.001` *Type*: Float
`-tolerance` *value*	Specifies the accuracy level of extractor in percent. For example, 5.0 represents a plus or a minus 5.0 percent allowable error in the extracted model. The higher the tolerance, the faster the extraction time. *Default:* `1.0` *Type:* Float
`-validation_dir` *string*	Enables the validation of generated ETM for corresponding view and displays number of failures at the end of validation. It is mandatory to specify the name of validation directory with this option, this contains the `report_timing_model`, `check_timing_model`, and summary reports.
`-view` *viewname*	Specifies the current active view.

`-verilog_shell_file` *`filename`*	Generates a Verilog shell file that instantiates the timing model. This parameter is useful for validating the model. Use the `-verilog_shell_module` parameter to specify the name of the top module in this Verilog file.
`-verilog_shell_module` *`top_module_name`*	Specifies the top module name of the Verilog shell file. *Default:* `test_shell`

Examples:

- The following command builds a `.lib` model (`block.lib`) for the block:

  ```
  write_timing_model block.lib
  ```

- The following command creates the timing model for the `test_model` cell belonging to the `test_lib` library, and writes the model into the `test.lib` liberty file:

  ```
  write_timing_model -cell_name test_model -library_name test_lib test.lib
  ```

Related Commands

- `check_timing_model`

42

Timing Optimization Commands and Attributes

- add_repeater_by_rule
- add_tieoffs
- add_tieoffs Category Attributes
- check_footprint
- create_basic_path_groups
- delete_buffer_trees
- delete_tieoffs
- eco_clone
- eco_clone_flop
- eco_merge_combinational_cell
- eco_split_combinational_cell
- eco_split_flop
- eco_split_complex_flop
- get_scheduling_file
- opt_area
- opt_clock_skew
- opt_design
- opt_power
- opt_signoff
- opt_signoff Category Attributes
- opt_virtual

- opt Category Attributes
- opt Category Object Attributes
- read_footprint
- report_always_on_buffer
- report_critical_nets
- report_critical_insts
- report_critical_pins
- report_footprints
- report_ignored_nets
- report_lef_equivalent_cells
- report_length_violations
- report_multibit
- report_multi_bit_ffs
- report_path_group_options
- report_route_type_constraints
- report_threshold_instance_count
- reset_path_group_options
- route_opt_design
- set_buffer_footprint
- set_delay_footprint
- set_inverter_footprint
- set_max_cap_per_freq
- set_max_cap_per_freq_transition
- set_max_transition_per_freq
- set_path_group_options
- set_scheduling_file
- write_multi_bit_flop_mapping_file

- write_target_based_opt_file

add_repeater_by_rule

```
add_repeater_by_rule
[-help]
[-copy_net_attribute]
[-exclude_nets_file exclude_net_file]
[-include_nets_file inlcude_net_file]
[-net_mapping file_name]
[-pre_route | -post_route | -along_route]
[-report_dir directory_name]
[-report_ignored_nets ignored_net_file]
[-rule file_name]
[-template]
```

Inserts buffers and/or inverters to nets, as defined in a rule file. By default, inserts available buffers and/or inverters with the smallest total area cost while satisfying the rules. As compared to DRV fixing, `add_repeater_by_rule` does not consider transition violations (maxTran) or any timing information. Instead, it constrains max total net length, max radius length, max capacitance, max fanout count, and so on. Also, `add_repeater_by_rule` does not legalize newly added buffers. You need to legalize the added buffers to a legal location.

The command provides an efficient, buffering method for top-level designs.

Note: Max radius length refers to the sum of wire lengths from the driver to the farthest sink and max total net length refers to the total (sum) wire lengths throughout the target net.

The command has the following features:

- Supports multiple buffer/inverter/AO cells simultaneously
- Supports multiple types of constrains simultaneously
- Provides complete MSV compatibility solution
- It is multithreaded

The rule file may contain two categories of constraints, where each category further breaks down to several constraint types. All constraints share the same syntax of ‘command value’.

1. Individual repeaters’ rule: constrains each repeater’s driving capability.

- max net total length (um): SetInverterMaxNetLength, SetBufferMaxNetLength, SetBufferMinLayerMaxNetLength, SetInverterMinLayerMaxNetLengt

h

- max radius length (um): SetInverterMaxRadiusLength, SetBufferMaxRadiusLength
- max capacitance (pF): SetInverterMaxCap, SetBufferMaxCap
- max fanout count: SetInverterMaxFanout, SetBufferMaxFanout
- max fanout at sink count: SetMaxFanoutAtSink
- max wirelength (um) of the net driving the sink term: SetMaxNetLengthAtSink

2. Global default rule: constrains all other instances beyond those newly inserted repeater instances.

- default max net total length (um): SetDefaultMaxNetLength, SetDefaultMinLayerMaxNetLength
- default max radius length(um): SetDefaultMaxRadiusLength
- default max cap length (pF): setDefaultMaxCap
- default max fanout count: setDefaultMaxFanout
- default max sink radius length (um): SetDistLastRepToSink
- default max wirelength (um) of the net driving the sink term: SetMaxNetLengthAtSink

Those constraints can be used individually or combined. However, at least one valid repeater must be specified in the rule file. If the global default rule for certain constraint type is omitted from the rule file, the command automatically picks the tightest constraint from the repeaters defined in the rule file under the ***same*** type of constraint as the default constraint. A sample rule file is shown in the "Example" section.

The following three layer-based rules are provided to support specifying net length constraints independently for each user specified layer:

- minimum layer for max net total length (um): SetBufferMinLayerMaxNetLength and SetInverterMinLayerMaxNetLength
- default minimum layer for max net total length (um): SetDefaultMinLayerMaxNetLength

Note: You must specify the bottom_preferred_routing_layer/top_preferred_routing_layer attribute for the nets that need layer-based buffering. If no attribute is present, the net will be routed with the default rule on default layers.

Use Model

When max layer constraint is specified:

```
SetBufferMaxNetLength BUF_X4M_A9TR 300.0
```

In the above scenario, ARR will buffer the net with `BUF_X4M_A9TR` at length intervals of `300um`.

When minimum layer support is also specified along with max layer constraint, then the following two scenarios are possible:

Example 1:

```
SetBufferMaxNetLength          BUF_X4M_A9TR     300.0
SetBufferMinLayerMaxNetLength  BUF_X4M_A9TR 6   500.0
```

In the above example, layers 2-5 will have a `300um` max length, and becasue there is a minimum layer constraint on the net of layer 6 or higher, the length constraint will be increased from `300` to `500um`.

Example 2:

When multiple minimum layer constraints are specified:

```
SetBufferMaxNetLength            BUF_X4M_A9TR     300.0
SetBufferMinLayerMaxNetLength    BUF_X4M_A9TR 6   500.0
SetBufferMinLayerMaxNetLength    BUF_X4M_A9TR 11  800.0
```

In the above example, layers 2-5 will have a `300um` max length; layers 6-9 will have a `500um` max length; and 11+ a `800um` max length.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `add_repeater_by_rule` parameter. For a detailed description of the command and all of its parameters, use `man add_repeater_by_rule` .
`-along_route`	Allows repeater insertion along the route. The database should be global-routed with pre-Route RC extraction before calling this option. **Note**: Run `place_detail` and `route_eco` if you specify this parameter. *Data_type*: bool, optional
`-copy_net_attribute`	When this parameter is specified, the net attributes set on the original nets are copied to the nets inserted by this command. *Data_type*: bool, optional
`-exclude_nets_file` *file_name*	

	Specifies a file that contains the names of hierarchical nets to exclude from rule violation repair. If a net appears in the files specified by both `-exclude_nets_file` and `-include_nets_file`, it is excluded from rule violation repair. *Data_type*: string, optional
`-include_nets_file` *file_name*	
	Specifies a file that contains the names of hierarchical nets considered by this command. If a net appears in the files specified by both `-exclude_nets_file` and `-include_nets_file`, it is excluded from rule violation repair. *Data_type*: string, optional
`-net_mapping` *file_name*	

	When specified, generates a file that maps the original net names to the nets inserted by this command. An example of the file format is provided below. In this example, `netA` is split into two nets; `FE_new_net1_netA` and `FE_new_net2_netA`.
	`netA FE_new_net1_netA`
	`netA FE_new_net2_netA`
	If, in the next iteration, `FE_new_net1_netA` is split again, the file would contain the following:
	`FE_new_net1_netA FE_new_net1_FE_new_net1_netA`
	`FE_new_net1_netA FE_new_net2_FE_new_net1_netA`
	Data_type: string, optional

`-post_route`	Allows repeater insertion along the route. The database should be detailed-routed with detailed RC extraction before calling this option. Preserves routing so that changes are minimized. **Note** : Run `place_detail` and `route_eco` if you specify this parameter. *Data_type*: bool, optional

`-pre_route`	Allows repeater insertion with different routing topologies. The database should be global-routed with pre-Route RC extraction before calling this option. When this parameter is specified, the software explores different typologies to optimally insert repeaters. *Data_type*: bool, optional
`-report_dir` *`directory_name`*	Specifies the directory where the software writes detailed failure reason report that is generated when this command runs. By default, the failure reason report is dumped in the `timingReports/ARRFailureReason/<design>` directory. For example, `timingReports/ARRFailureReason/<design>.arr.preRoute.rpt.gz` *Default*: `timingReports` *Data_type*: string, optional
`-report_ignored_nets` *`file_name`*	
	Reports the nets where no buffers were inserted and the reason the nets were ignored. *Data_type*: string, optional
`-rule` *`file_name`*	Specifies a file you create that should list the buffers and inverters and the length and capacitance rules for inserting each of them. Only buffers and inverters included in the rule file are inserted. *Data_type*: string, optional
`-template`	Writes a template rule file. **Note**: When the `-template` parameter is specified, the software does not honor any other parameters for this command. *Data_type*: bool, optional

Examples

- The following command explores different typologies to optimally insert repeaters:
  ```
  add_repeater_by_rule -rule rule1 -pre_route
  ```
- In a detailed routed database with RC extraction, the following command inserts repeaters along the route and preserves routing, so that changes are minimized:
  ```
  add_repeater_by_rule -rule rule1 -post_route
  ```
- The following command inserts repeaters on those nets specified in the file, `net_file1`:
  ```
  add_repeater_by_rule -rule rule1 -include_nets_file net_file1
  ```
- The following is an example of a template rule file:

```
##############################################################################
################### add_repeater_by_rule template rule file ###################
##############################################################################
# constraintType            cellName              constraint value
#                                                Length(um)/cap(pF)/count
##############################################################################

############ max net total Lengths for individual repeaters
  SetInverterMaxNetLength    HS65_LH_IVX7          150.0
  SetBufferMaxNetLength      HS65_LS_BFX7          300.0

############ max radius Lengths for individual repeaters
  SetInverterMaxRadiusLength HS65_LH_IVX7          100.0
  SetBufferMaxRadiusLength   HS65_LS_BFX7          240.0

############ max capacitance for individual repeaters
  SetInverterMaxCap          HS65_LH_IVX7          10.0
  SetBufferMaxCap            HS65_LS_BFX7          20.0

############ max fanout for individual repeaters
  SetInverterMaxFanout       HS65_LH_IVX7          20
  SetBufferMaxFanout         HS65_LS_BFX7          30

############ default max net total Lengths
  SetDefaultMaxNetLength                           100.0

############ default max radius Lengths
```

```
  SetDefaultMaxRadiusLength                          200.0

########### default max capacitance
  SetDefaultMaxCap                                   15.0

########### default sink max radius length
  SetDistLastRepToSink                               50.0

########### max net lengths from sink
  SetMaxNetLengthAtSink                             100.0

########### default max fanout
  SetDefaultMaxFanout                                20


#############################################################################

#   'DrivingStrength' constrains both max radius Length and max cap.
#############################################################################
#   constraintType              cellName            Length(um)      cap(pF)
#############################################################################

    SetInverterDrivingStrength  HS65_LH_IVX7        100.0           10.0
    SetBufferDrivingStrength    HS65_LS_BFX7        240.0           20.0
    SetDefaultDrivingStrength                       200.0           15.0
```

Related Topics

- place_detail
- route_eco
- write_repeater_rules
- The following sections in Optimizing Timing chapter of *Innovus Stylus Common UI User Guide*
  - Optimizing Timing Using a Rule File
  - Default Naming Conventions

add_tieoffs

```
add_tieoffs
[-help]
[-create_hport {true|false}]
[-keep_existing {true|false}]
[-lib_cell tieCellName+]
[-matching_power_domains {true|false}]
[-post_mask {true|false}]
[-power_domain powerDomainName]
[-prefix prefixName]
[-report_hier_ports {true|false}]
[-cell_pin cellName:cellPin | -instance_pin file_name | -exclude_pin file_name]
```

Adds instances of specified tie-off cells to the logical hierarchy of the design and connects the tie-off pins of netlist instances to the tie-off pins of the added instances. Thus, these instances provide the connectivity between the tie-high and tie-low pins of the netlist instances to power and ground. Use this command after placing the standard cells in the flow.

Note: If you do not specify parameters, this command uses the global parameters specified by the `add_tieoffs` attributes.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `add_tieoffs` parameter. For a detailed description of the command and all of its parameters, use the man command: `man add_tieoffs`.
`-cell_pin` *cellName:cellPin*	Specifies the cell pins that are to be tied. The cell format could be "`cellName1:cellPin cellName2:cellPin cellName3:cellPin`". Only the cell pins that have tie 0/1 connections and are specified in `-cell_pin` are changed to the tie cell. The left tie connections do not change. Default: `""` (empty string) *Data_type*: string, optional

`-create_hport {true | false}`	Allows the added tie cells to connect to tie pins across hierarchical boundaries if other constraints, such as maximum fanout and maximum distance, allow it. *Default:* `false` *Data_type*: bool, optional
`-exclude_pin` *file_name*	Specifies the excludePin file name. The file format is "instanceName/instancePin". Each line in the file contains one instance pin. Only the instance pins that have tie 0/1 connections and are not defined in this file are changed to the tie cell. The left tie connections do not change. Default: `""` (empty string) *Data_type*: string, optional
`-instance_pin` *file_name*	Specifies the instance pins file name. The file format is "instanceName/instancePin". Each line in the file contains one instance pin. Only the instance pins that have tie 0/1 connections and are defined in this file are changed to the tie cell. The left tie connections do not change. Default: `""` (empty string) *Data_type*: string, optional
`-keep_existing {true|false}`	When set to `true`, preserves the existing tie-offs, added in the previous run, in the design and adds new ones. *Default:* `false` *Data_type*: bool, optional
`-lib_cell` *tieCellName+*	Specifies the tie cells to add. You can only specify a maximum of two tie-cells, where one cell must be a tie-high driver, and the other a tie-low driver. The two tie-cell names must be surrounded by quotation marks. This parameter overrides cell name(s) specified by the add_tieoffs Category Attributes. Default: "" (empty string) *Data_type*: (base_cell)+, optional

`-matching_power_domains {true|false}`	Inserts tie cells in all power domains. *Data_type*: bool, optional
`-post_mask {true | false}`	Insert tie cells in post mask flow. Instructs the software to reuse existing tie cells to tie off a newly created spare instance in the design instead of adding or deleting tie cells. Searches for input pins that are tied to logical 1 or logical 0 and attempts to tie them off to tie high or tie low cells. *Default:* `false` *Data_type*: bool, optional
`-power_domain` *powerDomainName*	Inserts tie-off cell to the named PD. Default: `""` (empty string) **Note:** When this option is not specified, tool only inserts tie cells in the "default" power domain. *Data_type*: string, optional
`-prefix` *prefixName*	Specifies a prefix for the added tie-cell instances. Default: `""` (empty string) *Data_type*: string, optional
`-report_hier_ports {true | false}`	When the value of `-create_hport`is `true`, reports created port connections to a file named `tiehilo.rpt`. *Default:* `false` *Data_type*: bool, optional

Examples

- The following command adds tie-high and tie-low connectivity for designs with libraries that have cell named TIEOFF that provides both tie-high and tie-low connections. The instances added are prefixed with tieOff:

  ```
  add_tieoffs -lib_cell TIEOFF -prefix tieOff
  ```

- The following command adds tie-high and tie-low connectivity for designs with libraries that have separate tie-high and tie-low cells named TIEHI and TIELO respectively:

```
add_tieoffs -lib_cell "TIEHI TIELO"
```

- The following commands add tie-off connectivity for MSV designs with two power domains named `PD1` and `PD2`. The last command is necessary for the part of the design that is not in either of the two specified power domains.

```
add_tieoffs -lib_cell "TIEHI TIELO" -power_domain PD1
add_tieoffs -lib_cell "TIEHI TIELO" -power_domain PD2
add_tieoffs -lib_cell "TIEHI TIELO"
```

Related Information

- `check_tieoffs`
- `delete_tieoffs`
- The following section in Optimizing Timing chapter of *Innovus Stylus Common UI User Guide*
  - Adding Logical Tie-Off Cells

add_tieoffs Category Attributes

The root attributes in the add_tieoffs category are:

- add_tieoffs_cells
- add_tieoffs_create_hports
- add_tieoffs_honor_dont_touch
- add_tieoffs_honor_dont_use
- add_tieoffs_max_distance
- add_tieoffs_max_fanout
- add_tieoffs_module_prevention
- add_tieoffs_prefix
- add_tieoffs_report_hports

To set the value of any root attribute, use the `set_db` command:

```
set_db attribute_name value
```

To get the current value of any root attribute, use the `get_db` command:

```
get_db attribute_name
```

To get the current value of all attributes in the `add_tieoffs` category, use the following command:

```
get_db -category add_tieoffs
```

add_tieoffs_cells

```
add_tieoffs_cells {{tieHi1 tieLo1} {tieHi2 tieLo2} ...}
```

Default: `""`

Data_type: string

Specifies the pairs of tie cells to be used by the `add_tieoffs` command. You can specify a maximum of two tie-cells in a cell list, where one cell must be a tie-high driver, and the other a tie-low driver.

A single cell with both tie connections, high and low, can be specified as below.

```
add_tieoffs_cells {{TieHiLo1} {TieHiLo2} {TieHiLo3} ….}
```

The software selects the nearest cell pair to make new tie connections during the post_mask flow.

Related Command

- `add_tieoffs`

add_tieoffs_create_hports

`add_tieoffs_create_hports {true | false}`

Default: `false`

Data_type: bool, read/write

Allows the added tie cells to connect to tie pins across hierarchical boundaries if other constraints, such as maximum fanout and maximum distance, allow it.

add_tieoffs_honor_dont_touch

`add_tieoffs_honor_dont_touch {true | false}`

Default: `false`

Data_type: bool, read/write

Allows both `add_tieoffs` and `add_tieoffs` commands to honor the `dont_touch` attribute set on instances, nets, cells, and modules.

Related Commands

- `add_tieoffs`
- `delete_tieoffs`

add_tieoffs_honor_dont_use

`add_tieoffs_honor_dont_use {true | false}`

Default: `false`

Data_type: bool, read/write

Honors `dont_use` attribute on cells.

add_tieoffs_max_distance

`add_tieoffs_max_distance` *`distanceValue`*

Default: `0`

Data_type: double, read/write

Specifies the maximum distance, in microns, between the tie-cell driver and the tie-pins. A value of `0` implies no distance constraint.

add_tieoffs_max_fanout

`add_tieoffs_max_fanout` *`fanOutValue`*

Default: `0`

Data_type: int, read/write

Specifies the number of tie-pins a tie-net can drive. A value of `0` implies no fanout constraint.

Example

The following set of commands set `TIEHI` and `TIELO` cells a maximum distance of 20 microns between the tie-cell pin and the tie-off input pins and a tie-net can drive up to 10 tie-pins:

```
set_db add_tieoffs_max_fanout 10
set_db add_tieoffs_max_distance 20
set_db add_tieoffs_cells "TIEHI TIELO"
```

add_tieoffs_module_prevention

`add_tieoffs_module_prevention {true | false}`

Default: false

Data_type: bool, read/write

When this attribute is set to `true`, the `add_tie_offs` command adds tie-cells for module instantiation port. Tie-cells may also be added inside the module to tie the output port of the module so that module function can be prevented.

add_tieoffs_prefix

`add_tieoffs_prefix` *`prefixName`*

Default: `""`

Data_type: string, read/write

Specifies a prefix for the added tie-cell instances.

add_tieoffs_report_hports

`add_tieoffs_report_hports {true | false}`

Default: `false`

Data_type: bool, read/write

Report created ports when `add_tieoffs_create_hports` is set. Report is saved to a file named `tiehilo.rpt`.

check_footprint

```
check_footprint
[-help]
```

Checks the timing library for missing footprint functions or timing information.

To view the above information in the software, use the man command: `man check_footprint`.

Related Topics

- `read_footprint`
- `report_footprints`
- `set_buffer_footprint`
- `set_delay_footprint`
- `set_inverter_footprint`

- The following section in Optimizing Timing chapter in the *Innovus Stylus Common UI User Guide*
    - Using Cell Footprints

create_basic_path_groups

```
create_basic_path_groups
[-help]
[-expanded]
[-reset]
```

Explicitly creates standard path groups irrespective of the design stage. The following path groups are created:

- reg2reg (register-to-register)
- reg2cgate (register-to-clock-gate)
- default

Note:

- If a design does not have any reg2reg paths, or reg2cgate paths, this command will create the corresponding path group(s).
- If the specified path group, say `reg2reg`, already exists, the software will give the following warning:

  ```
  **WARN: (IMPOPT-3666): Path group reg2reg already exists. Skipping the creation of
  this path group.
  ```

Parameters

`-help`	Outputs a brief description that includes type and default information for each `create_basic_path_groups` parameter. For a detailed description of the command and all of its parameters, use the man command: `man create_basic_path_groups`.
`-expanded`	When this parameter is specified, the following path groups are created: • reg2reg • reg2cgate • in2reg • reg2out, and • in2out When this parameter is not specified, only the default path groups, reg2reg and reg2cgate, are created. **Note**: Timing optimization will consider these path groups in the same way as paths grouped using the `group_path` command. *Data_type*: bool, optional
`-reset`	Deletes all existing path groups. This is equivalent to running the `reset_path_group` command. **Note**: After deleting the existing path groups, new path groups, reg2reg and reg2cgate are created. *Data_type*: bool, optional

Examples

- The following command is used to create the standard path groups and assigns an effort level to each group:

  ```
  create_basic_path_groups
  ```

  The software returns the following information:

  ```
  Created reg2reg path group
  Effort level <high> specified for reg2reg path_group
  Created reg2cgate path group
  Effort level <high> specified for reg2cgate path_group
  ```

- The following commands are used to set the effort level for the reg2reg path group to high, and report the parameters of the path groups that are created:

  ```
  set_path_group_options reg2reg -effort_level low
  report_path_group_options
  ```

  The software prints the following report:

  ```
  ------------------------------------------------------------------------------
  -------------
  |Path Group|Effort|Adjustment(early/late)|Target
  Slack|weight|SkewingSlackCons|view          |
  ------------------------------------------------------------------------------
  -------------
  |reg2reg   | high | 0.1    /-0.2         | 0          |  2   |    0.3
  |AV_HL_FUNC_MIN_RC1 |
  |reg2cgate | high | 0.2    /0            | 0          |  1   |    0.5
  |AV_HL_FUNC_MAX_RC1 |
  ```

Related Information

- `group_path`
- `report_path_group_options`
- `reset_path_group`
- `reset_path_group_options`
- `set_path_group_options`
- The following section in Optimizing Timing chapter of *Innovus Stylus Common UI User Guide*
  - Using Additional Post-CTS Timing Optimization Parameters

delete_buffer_trees

```
delete_buffer_trees
[-help]
[-exclude_nets_file excfilename]
[-footprint footprintname]
[-nets_file netsfilename | -nets list_of_nets]
[-preserve_route]
[-verbose]
```

Removes all buffers and inverter pairs, of all footprint types, on all nets in the design. With specific settings, this command can remove buffers and inverter pairs of the specified footprint type, or on specified nets only.

Note:

1. place_opt_design will automatically call `delete_buffer_trees` in the beginning, before global placement.
2. If you want to have a clean design with reduced density, or just clear all buffers and inverter pairs of a certain list of nets, it is recommended to call this command. This may allow you to do ECO on those nets.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `delete_buffer_trees` parameter. For a detailed description of the command and all of its parameters, use the man command: `man delete_buffer_trees`
`-exclude_nets_file` *excfilename*	Specifies the input file that contains the names of nets to exclude from buffer removal. If a net appears in the files specified by both `-exclude_nets_file` and `-nets_file`, it is excluded. *Data_type*: string, optional
`-footprint` *footprintname*	Specifies the footprint of buffers to remove. If you do not specify this parameter, the Innovus software removes buffer instances with one input and one output terminal, and whose timing arc is not `SPECIAL`. *Data_type*: string, optional
`-nets` *list_of_nets*	Specifies the hierarchical nets to include during buffer removal. **Note**: The `-nets` and `-nets_file` parameters are mutually exclusive. *Data_type*: string, optional
`-nets_file` *netsfilename*	Specifies the file that contains the names of hierarchical nets to include during buffer removal. The Innovus software excludes all other nets. If a net appears in the files specified by both `-exclude_nets_file` and `-nets_file`, it is excluded. **Note**: The `-nets` and `-nets_file` parameters are mutually exclusive. *Data_type*: string, optional
`-preserve_route`	Specifies that nets not impacted by the buffer/inverter deletion process must have their routing unchanged. By default, the tool removes all the routes after running this command. *Data_type*: bool, optional
`-verbose`	When specified, displays additional debug messages in log file. For example, the failure reason for deletion of buffer and inverter pairs. *Data_type*: bool, optional

Example

- The following command Removes all buffers and inverter pairs, of all footprint types:

```
delete_buffer_trees
*** Start delete_buffer_trees ***
Info: Detect buffers to remove automatically.
Analyzing netlist ...
Updating netlist

*summary: 79 instances (buffers/inverters) removed
*** Finish deleteBufferTree (0:00:09.6) ***
```

- The following command removes all buffers of footprint type B_1P:

```
delete_buffer_trees -footprint B_1P
```

Related Information

- Optimizing Timing chapter in the *Innovus Stylus Common UI User Guide*
  - Using the Footprintless Flow

delete_tieoffs

```
delete_tieoffs
[-help]
[-cells tieCellName | "listOfTieCellNames"]
[-prefix prefixName]
```

Deletes placed tie cells from the netlist and reconnects the tie input pins back to `1'b1` and `1'b0`. Use this command if there are existing tie cells in design.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `delete_tieoffs` parameter. For a detailed description of the command and all of its parameters, use the man command: `man delete_tieoffs`
`-cells` *tieCellName* | "*listOfTieCellNames*"	
	The *tieCellName* argument specifies the cell name to delete. The "*listOfTieCellNames*" argument specifies the list of cell names to be deleted, which must be surrounded by quotation marks. **Note**: The *tieCellName* and "*listOfTieCellNames*" arguments are mutually exclusive. *Data_type*: string, optional
`-prefix` *prefixName*	Specifies the prefix of the instances to delete. *Data_type*: string, optional

Note: If you do not specify any parameters, the command uses the global cell names if they were set previously with the `add_tieoffs` Category attributes.

Example

- The following command deletes all placed tie-off cell instances with the prefix `ABC`:
  ```
  delete_tieoffs -prefix ABC
  ```
- The following command deletes all placed tie-off cells named: `TIEOFF`:

```
delete_tieoffs -cells TIEOFF
```

Related Information

- `add_tieoffs`
- `check_tieoffs`
- The following section in Optimizing Timing chapter of *Innovus Stylus Common UI User Guide*
  - Adding Logical Tie-Off Cells

eco_clone

```
eco_clone
[-help]
[-allow_sequential]
[-auto]
[-balance num]
[-cells list_of_cells]
[-delete_buffer_trees]
[-dont_legalize]
[-dont_resize]
[-force]
[-from pin names]
[-instances instance names]
[-locs list_of_locations]
[-loop number]
[-max_levels integer]
[-max_num_loads integer]
[-min_num_loads integer]
[-nets list_of_nets]
[-num_clones number]
[-pins list_of_term_lists]
[-preview]
[-radius number]
[-same_hier_only]
[-to pin names]
[-verbose]
```

Clones combinational logic if logic does not have any user constraints like don’t size/touch and so on. This command is used when you want to clone combinational logic as manual ECO. It does not work for sequential logic like flops. The `-from` and `-to` parameters are used to specify the 'from' and 'to' pins for cloning.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `ecoClone` parameter. For a detailed description of the command and all of its parameters, use the man command: `man ecoClone`

`-allow_sequential`	Used with `-auto` to try cloning at the sequential gates too. *Data_type*: bool, optional
`-auto`	Applies cloning and buffering automatically. The software identifies and clones combinational logic based on critical endpoints distance-based clustering. *Data_type*: bool, optional
`-balance` *number*	When specified, forces the number of loads on each clone to be within the specified number percentage (*num %*). The default balance is 10%. *Data_type*: integer, optional
`-cells` *list_of_cells*	Specifies the list of cells to use for the clones. **Note**: ◦ If this parameter is not specified, the clones have the same cell as the original gate. ◦ If only one cell is specified in the list, all the clones are created with that cell. ◦ If the `-locs` and `-terms` parameters are not specified, you can only specify one cell in that list. ◦ If the `-locs` and/or `-terms` parameters are specified, you can also specify numClones cells, which is one cell for each clone. ◦ It is not possible to specify a new cell for the original gate, so the original gate is not resized. *Data_type*: string, optional
`-delete_buffer_trees`	When specified, the software attempts to delete buffer trees driven by instances to clone. **Note**: The deletion of buffer trees with user constraints is skipped. *Data_type*: bool, optional

`-dont_legalize`	When specified, the software may leave cloned instances overlapped with the original ones. This means the software will not try to legalize instances. This means that if space is available, it will place legally, otherwise it will leave cloned instances overlapped with the original ones. *Data_type*: bool, optional
`-dont_resize`	Specifies that neither the original nor the new (cloned) instances should be resized. *Data_type*: bool, optional
`-force`	When specified, forces instance cloning, even if it worsens QoR or causes illegal placement. *Data_type*: bool, optional
`-from` *pin names*	Specifies 'from' pins. You can specify a single pin or a list of pins. *Data_type*: string, optional
`-instances` *instance_names*	Specifies the instances to be cloned. You can specify a single instance or a tcl list of instances to be cloned. **Note**: The `-nets` and `-instances` parameters are mutually exclusive. *Data_type*: string, optional
`-locs` *list_of_locations*	Specifies the location of each clone. This parameter is used along with the `-num_clones` parameter. When this parameter is specified, a location should be specified for each clone, so the length should be numClones. The length can be numClones+1, in which case, the first location is used for the original gate. When the locations are specified, the sink partitions are assigned to the closest clone. *Data_type*: string, optional
`-loop` *number*	Repeat cloning at most *number* times, as far as new instances are being created. The original and new instances within the min/max number of loads range are valid targets at each iteration. *Default*: `0` (means no limit) *Data_type*: integer, optional

`-max_levels` *integer*	Specifies the level of instances that the software needs to traverse for cloning. Zero value means the software will clone everything till the origin of logic. *Default*: 0 (meaning infinite) *Data_type*: integer, optional
`-max_num_loads` *integer*	Specifies that cloning should be skipped for instances driving more than the specified number of loads. *Default*: `50000` *Data_type*: integer, optional
`-min_num_loads` *integer*	Specifies that cloning should be skipped for instances with less than or equal to the specified number of loads. **Note**: The `-min_num_loads` and `-num_clones` parameters are mutually exclusive. *Data_type*: integer, optional
`-nets` *list_of_nets*	Specifies the nets to be cloned. You can specify a single net or a tcl list of nets to be cloned. **Note**: The `-nets` and `-instances` parameters are mutually exclusive. Wildcards are allowed for this parameter. *Data_type*: string, optional
`-pins` *list_of_term_lists*	Specifies the partitions, so it is a list of term lists. This parameter is used along with the `-numClones` parameter. When this parameter is specified, there should be one partition for each clone, so the length should be numClones. The terms that are not in any partition are assigned to the partition driven by the original gate. The length can be numClones+1, in which case the first partition is for the original gate. In that case, all the sinks should be in a partition. *Data_type*: string, optional

`-preview`	When specified, provides a preview of what would be done, while leaving the netlist untouched. This means that the software will report instances that will be cloned and the changes that will be applied to the netlist without actually changing the netlist. *Data_type*: bool, optional
`-num_clones` *number*	Specifies the number of clones to create. The default value is `0` and in that case, it is computed from the `-min_num_loads` parameter. **Note**: The `-min_num_loads` and `-num_clones` parameters are mutually exclusive. *Default:* `0` *Data_type*: integer, optional
`-radius` *number*	Specifies that the clones should be placed within a radius of the specified number of rows from their preferred location based on the original instance location and the center of mass of the new set of loads. *Default*: `10` *Data_type*: integer, optional
`-same_hier_only`	Specifies that only those loads will be considered that are in the same hierarchy as the driver. *Data_type*: bool, optional
`-to` *pin names*	Specifies 'to' pins. You can specify a single pin or a list of pins. *Data_type*: string, optional
`-verbose`	When specified, verbose messages are displayed during the cloning of instances. *Data_type*: bool, optional

Example

The following command will clone logic till `$to` pins with more then `6` loads, while forciby deleting the buffer tree:

```
eco_clone -to $to -verbose  -min_num_loads 6 -delete_buffer_trees -force
```

Related Information

- design Category Attributes
- `eco_clone_flop`
- `eco_split_flop`
- opt Category Attributes
- The following section in Optimizing Timing chapter of *Innovus Stylus Common UI User Guide*
  - Using the Multi-Bit Flip-Flop Flow

eco_clone_flop

```
eco_clone_flop
[-help]
[-force]
-instances flop_instance_name
[-max_num_loads num_of_loads]
[-num_clones number]
[-radius num_of_rows]
[-verbose]
```

Clones flop instances as long as the flops meet the following criteria:

- They are not FIXED
- They are allowed to be resized
- When they do not share synchronization signals with some other instance

This command is used when you want to clone a flop as manual ECO. Manual flop cloning is usually performed on control signal flip-flops, when the output needs to reach sections of the floorplan that are spread out. Cloning the flip-flops allows you to place each clone close to its sink points, thereby reducing the timing closure effort.

This command should not be used to clone a flop that drives a clock net, which is a net with the attribute `isDefInclock`, because this may lead to problems during optimization or timing. The default behavior of the command may reject the flip-flop cloning due to design constraints or QOR impact.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `eco_clone_flop` parameter. For a detailed description of the command and all of its parameters, use the man command: `man eco_clone_flop`
`-force`	When specified, forces the flop instance to clone. This parameter is used when flop cloning is rejected by the software. *Data_type*: bool, optional
`-instances` *flop_instance_name*	Specifies the list of flop instances to be cloned. This is a required parameter. *Data_type*: string, optional
`-max_num_loads` *num_of_loads*	Specifies that cloning should be skipped for instances driving more than the specified number of loads. *Default*: `50000` *Data_type*: int, optional
`-num_clones` *number*	Specifies the cloned numbers for the flop instance. *Data_type*: int, optional
`-radius` *num_of_rows*	Specifies that the clones should be placed within a radius of the specified number of rows from their preferred location based on the original instance location and the center of mass of the new set of loads. *Default*: `10` *Data_type*: int, optional
`-verbose`	When specified, verbose messages are displayed during the cloning of flop instances. *Data_type*: bool, optional

Example

The following command will clone the flip-flop `$inst` 3 times:

```
eco_clone_flop -instances $inst -num_clones 3
```

Related Information

- design Category Attributes
- `eco_split_flop`
- `eco_split_complex_flop`
- `report_multi_bit_ffs`
- `write_multi_bit_flop_mapping_file`
- The following section in Optimizing Timing chapter of *Innovus Stylus Common UI User Guide*
  - Using the Multi-Bit Flip-Flop Flow

eco_merge_combinational_cell

```
eco_merge_combinational_cell
[-help]
[-area]
[-force]
-inst comboInstName | -list_file instNamesFile | -all
[-power]
-pre_cts | -post_cts
```

Merges combinational cells into multi-bit cells for the specified instance. You can specify the names of the combinational instances to be merged or you can specify the file containing names of single bit instances to merge.

Naming Convention During Merging

When the combinational cells in the specified file are merged, the corresponding instance names are combined as follows:

- `hier1/hier2/i_0 + hier1/hier2/i_1 + hier1/hier2/i_2 + hier1/hier2/i_3`
  TO
  `hier1/hier2/CDN_MB_i_0_MB_i_1_MB_i_2_MB_i_3`
- `abc_x[0] + abc_x[1] + abc_x[2] + abc_x[3]`
  TO
  `CDN_MB_abc_x[0]_MB_x[1]_MB_x[2]_MB_x[3]`

Parameters

`-help`	Outputs a brief description that includes type and default information for each `eco_merge_combinational_cell` parameter. For a detailed description of the command and all of its parameters, use the man command: `man eco_merge_combinational_cell`.
`-all`	Specifies that all single-bit cell instances should be merged into multi-bit cell instances. This parameter is mutually exclusive with the `-list_file` and `-inst` parameters. At least one of these parameters must be specified.

`-area`	Specifies to merge the combinational cells for area purposes. *Default*: `false`
`-force`	Forces the merging of the combinational cell regardless of the timing gains. *Default*: `false`
`-inst` *`comboInstName`*	Specifies the name of the combinational instance to be merged. This is a required parameter. This parameter is mutually exclusive with the `-list_file` and `-all` parameters. At least one of these parameters must be specified.
`-list_file` *`instNamesFile`*	Specifies the file containing the names of single bit instances to merge. Specify one instance on one line. Comment with Instance name should not contain spaces or backslashes This parameter is mutually exclusive with the `-inst` and `-all` parameters. At least one of these parameters must be specified.
`-post_cts`	Specify this parameter if the design stage is post-CTS. **Note**: This parameter is mutually exclusive with the `-pre_cts` parameter. One of the two parameters must be specified.
`-power`	Specify this parameter to merge the combinational cells for power purposes. *Default*: `false`
`-pre_cts`	Specify this parameter if the design stage is pre-CTS. This is a required parameter. **Note**: This parameter is mutually exclusive with the `-post_cts` parameter. One of the two parameters must be specified.

Related Information

- The following chapter of *Innovus Stylus Common UI Text Command Reference*
  - opt Category Attributes
- `eco_clone_flop`
- `eco_split_combinational_cell`
- `eco_split_complex_flop`
- `report_multi_bit_ffs`
- `write_multi_bit_flop_mapping_file`
- The following section in Optimizing Timing chapter in the *Innovus Stylus Common UI User Guide*
  - Multi-Bit Combinational Cell Optimization

eco_split_combinational_cell

```
eco_split_combinational_cell

[-help]
[-force]
[-full_split]
[-legal]
[-skip_route]
-inst combInstName | {{Mbcb1 combCell1} {Mbcb2 combCell2}...} | -list_file fileName | -all
-pre_cts | -post_cts | -post_route
```

Splits all multi-bit combinational cell instances to single-bit combinational cells. You can specify the names of the combinational instances to be merged or you can specify the file containing names of multi-bit instances to be split.

Naming Convention During Splitting

When splitting Multibit combinational cells, it will split the names as follows:

- `hier1/hier2/i_[0:3]`
  TO
  `hier1/hier2/i_[0:3]__0, hier1/hier2/i_[0:3]__1, hier1/hier2/i_[0:3]__2, hier1/hier2/i_[0:3]__3`
- `CDN_MB_abc_MB_def`
  TO
  `abc, def`

Parameters

`-help`	Outputs a brief description that includes type and default information for each `eco_split_combinational_cell` parameter. For a detailed description of the command and all of its parameters, use the man command: `man eco_split_combinational_cell`.

`-all`	Splits all multi-bit combinational cell instances to SBFF instances. This parameter is mutually exclusive with the `-inst` and `-list_file` parameters. At least one of these parameters must be specified. *Default*: `false`
`-force`	Forces the splitting of the combinational cell regardless of the timing gains. *Default*: `false`
`-full_split`	Set this parameter to completely split the multi-bit combinational cells to single bits.
`-inst` *`comboInstName`* `{{`*`Mbcb1 combCell1`*`} {`*`Mbcb2 combCell2`*`}...}`	
	Specifies the names of the combinational instance to be split. If the input is a string of multi-bit combinational (MBCB)/Cell pairs list, it will split the MBCB to the specified cell without evaluation This parameter is mutually exclusive with the `-list_file` and `-all` parameters. At least one of these parameters must be specified.
`-legal`	Specifies to place the split flip-combinational cells at legal location. *Default*: `false`
`-list_file` *`fileName`*	Specifies the file containing names of MBCB instances to split. Specify one instance on one line. Comment with #. Instance name should not contain spaces or backslashes. And if one instance and one cell is specified on one line, it will split the MBCB to the specified cell without evaluation. This parameter is mutually exclusive with the `-inst` and `-all` parameters. At least one of these parameters must be specified.
`-post_cts`	Specify this parameter if the design stage is post-CTS. This is a required parameter. One of these parameters, `-pre_cts`, `-post_cts`, or `-post_route` must be specified.
`-post_route`	Specify this parameter if the design stage is postRoute. This is a required parameter. One of these parameters, `-pre_cts`, `-post_cts`, or `-post_route` must be specified.

`-pre_cts`	Specify this parameter if the design stage is preCTS. This is a required parameter. One of these parameters, `-pre_cts`, `-post_cts`, or `-post_route` must be specified.
`-skip_route`	Specify this parameter if the design stage is post-Route and global detail routing is not required after the split.

Related Information

- The following chapters of *Innovus Stylus Common UI Text Command Reference*
  - opt Category Attributes
- `eco_clone_flop`
- `eco_merge_combinational_cell`
- `eco_split_complex_flop`
- `report_multi_bit_ffs`
- `write_multi_bit_flop_mapping_file`
- The following section in Optimizing Timing chapter in the *Innovus Stylus Common UI User Guide*
  - Multi-Bit Combinational Cell Optimization

eco_split_flop

```
eco_split_flop
[-help]
[-batch]
[-force]
[-full_split]
[-legal]
[-power]
{-inst flopInstName | {{Mbff1 flopCell1} {Mbff2 flopCell2}...} | -list_file <fileName>
| -all}
{-pre_cts | -post_cts | -post_route}
[-skip_route]
```

Splits a multi-bit flip-flop (MBFF) into single-bit flip-flops for the specified instance. By default, a specific flip-flop is split only if equivalent single-bit flip-flops can be found in libraries and there is timing gain after the split. You can also specify that the split flip-flops should be placed in legal locations. By default, the positions of the split flip-flops are not guaranteed to be legal.

Note:

- This command is not controlled by the `opt_multi_bit_flop_opt` attribute setting of opt Category Attributes
- The command calls refinePlace at the end. To disable this, set the `eco_refine_place` attribute to `false`
- MBFFs must be identified by the the command `report_multi_bit_ffs -all`
- A placeable location for the split FFs is required

Parameters

`-help`	Outputs a brief description that includes type and default information for each `eco_split_flop` parameter. For a detailed description of the command and all of its parameters, use the man command: `man eco_split_flop`.

`-all`	When specified, splits all MBFFs into smaller-bit flip-flops including single-bit flip-flops (SBFFs). Only timing critical bits are split into single-bit flops, and the remaining, non-critical bits, are split into smaller bit width MBFFs, to preserve power as much as possible. For example, a 4-bit MBFF may be split into two single-bit flops and one 2-bit flop. **Note**: The `-all`, `-inst` and `-list_file` parameters are mutually exclusive. If the `-all` parameter is not specified, one of these two parameters, `-inst` or `-list_file`, must be specified. *Default*: `false` *Data_type*: bool, optional
`-batch`	When specified, splits the MBFFs in batches. For example, `eco_split_flop -list_file list -post_cts -batch -force` *Default*: `false` *Data_type*: bool, optional
`-force`	Forces the flip-flop splits irrespective of timing gains. *Default*: `false` **Note**: • If there is no timing gain and this parameter is not specified, the flip-flops for the specified instance will not be split. • If this option is set along with the `-legal` option, the software will first look for legal locations. However, the flip-flops will be split even if no legal location is found. *Data_type*: bool, optional
`-full_split`	Forces the MBFFs to be completely split into all single-bit flip-flops. *Data_type*: bool, optional
`-inst flopInstName | {{Mbff1 flopCell1} {Mbff2 flopCell2}...}`	

	Specifies pairs of MBFFs to be split and the corresponding single-bit flops to be used. You can also specify a single instance or a tcl list of instances to be split. **Note**: The `-all`, `-inst`, and `-list_file` parameters are mutually exclusive. If the `-all` parameter is not specified, one of these two parameters, `-inst` or `-list_file`, must be specified. *Data_type*: string, required
`-list_file` *fileName*	
	Specifies the file containing pairs of MBFFs to be split and the corresponding single-bit flops to be used. You can also just specify the names of MBFF instances to be split. You can specify up to 50,000 instances to split. Specify each instance on a line and comment with #. Instance name should not contain spaces or backslashes. **Note**: The `-all`, `-inst`, and `-list_file` parameters are mutually exclusive. If the `-all` parameter is not specified, one of these two parameters, `-inst` or `-list_file`, must be specified. *Data_type*: string, required
`-legal`	Places the split flip-flops at legal locations. **Note**: If both the `-legal` and `-force` parameters are specified, the software will first look for legal locations. However, the flip-flops will be split even if no legal location is found. In this case, the split instances will be placed in the original location of the MBFFs. *Default*: `false` *Data_type*: bool, optional
`-power`	Splits the flip-flop for power purpose. *Default*: `false` *Data_type*: bool, optional
`-pre_cts`	Specify this parameter, if the design stage is pre-CTS. Default: `-post_route` *Data_type*: bool, optional

`-post_cts`	Specify this parameter, if the design stage is post-CTS. Default: `-post_route` *Data_type*: bool, optional
`-post_route`	Specify this parameter, if the design stage is post-Route. This is the default setting. *Data_type*: bool, optional **Note**: One of the three options, `-pre_cts, -post_cts,` or `-post_route` must be specified.
`-skip_route`	Specify if the stage is `post_route` and globalDetailRoute is not required to be performed after the split. Since `post_route` is also the default stage, globalDetailRoute is also skipped when the stage is default and `-skip_route` is specified. However, for the default case, when the stage is postRoute but this parameter is not specified, after splitting the instances, globalDetailRoute will be called. *Default*: `false` *Data_type*: bool, optional

Example

- The following command will force the flip-flop split for the instance, `ff_0`:

```
eco_split_flop -inst ff_0 -force
```

Related Information

- The following chapters of *Innovus Stylus Common UI Text Command Reference*
  - design Category Attributes
  - eco Category Attributes
  - opt Category Attributes
- `eco_clone_flop`
- `eco_split_complex_flop`
- `report_multi_bit_ffs`
- `write_multi_bit_flop_mapping_file`
- The following section in Optimizing Timing chapter in the *Innovus Stylus Common UI User Guide*
  - Using the Multi-Bit Flip-Flop Flow

eco_split_complex_flop

```
eco_split_complex_flop
[-help]
[-force]
{-inst {inst1, inst2,...} | -list_file instNamesFile | -all}
[-legal]
[-power]
[-pre_cts | -post_cts | -post_route]
[-skip_route]
```

Splits a complex flip-flop into a simple flip-flop and combinational logic for the specified instance. A complex flip-flop means the D side has some combinational logic, such as AND, OR, MUX, AOI, and so on. By default, a specific flip-flop is split only if equivalent simple flip-flops and combinational logic can be found in libraries and there is timing gain or power gain after the split.

You can also specify that the split flip-flops should be placed in legal locations. By default, the positions of the split flip-flops are not guaranteed to be legal.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `eco_split_complex_flop` parameter. For a detailed description of the command and all of its parameters, use the man command: `man eco_split_complex_flop`
`-all`	Specifies that all MBFF instances should be split to SBFF instances. *Default*: `false` **Note**: The `-inst`, `-listFile`, `-all`, and `batch` parameters are mutually exclusive. One of these must be specified. *Data_type*: bool, optional
`-force`	When specified, forces the complex flip-flops to split regardless of the timing gains. *Data_type*: bool, optional
`-inst {inst1, inst2,...}`	

	Specifies the names of the flop instances to be split. You can specify a single instance or a tcl list of instances to be split. **Note**: The `-inst`, `-listFile`, and `-all` parameters are mutually exclusive. One of these must be specified. *Data_type*: string, required
`-legal`	When specified, places the split flip-flop at a legal location. *Data_type*: bool, optional
`-list_file` *instNamesFile*	
	Specifies the file containing the names of the MBFF instances to split. Specify one instance on one line. Comment with #. Instance name should not contain spaces or backslashes. **Note**: The `-inst`, `-listFile`, and `-all` parameters are mutually exclusive. One of these must be specified. *Data_type*: string, required
`-pre_cts`	Specify this parameter, if the design stage is pre-CTS. *Data_type*: bool, optional
`-post_cts`	Specify this parameter, if the design stage is post-CTS. *Data_type*: bool, optional
`-post_route`	Specify this parameter, if the design stage is post-Route. *Data_type*: bool, optional
`-power`	When specified, complex flops are split to reclaim power. By default, the flops are split for timing. *Data_type*: bool, optional
`-skip_route`	Skip globalDetailRoute after split in post_route stage. *Data_type*: bool, optional

Examples

- Use the following command to force the flip-flop split for power for all instances during the pre_cts stage.

  ```
  eco_split_complex_flop -force -pre_cts -power -all
  ```

- Use the following command for flip-flop split for timing for the instance ff1 during the pre_cts stage.

  ```
  eco_split_complex_flop -inst ff1 -pre_cts
  ```

Related Information

- `eco_split_flop`
- `eco_clone_flop`
- Optimizing Timing chapter of *Innovus Stylus Common UI User Guide*

get_scheduling_file

```
get_scheduling_file
[-help]
```

Returns the name of the current scheduling file used by the `opt_clock_skew` command. If no scheduling file name is set, `get_scheduling_file` returns nothing. If you have already run `opt_clock_skew`, this command reports the default file name. If you have specified or changed the scheduling file name using `set_scheduling_file`, then this command returns this name.

To view the above description in the software, use `man get_scheduling_file`.

Related Information

- `opt_clock_skew`
- `set_scheduling_file`
- The following section in Optimizing Timing chapter of *Innovus Stylus Common UI User Guide*
  - Performing Optimization Before Clock Tree Synthesis

opt_area

```
opt_area
[-help]
[-hold_aware]
[-no_declone]
[-no_delete_buffer]
[-no_down_size]
```

Creates space in the design by downsizing and deleting buffers, without worsening the slack or increasing the design rule violations.

Note: Cadence recommends using the `opt_design` command to correct timing violations during timing optimization instead of using this command. For information on using `opt_design`, see Optimizing Timing in the *Innovus stylus Common UI User Guide* .

Parameters

`-help`	Outputs a brief description that includes type and default information for each `opt_area` parameter. For a detailed description of the command and all of its parameters, use the man command: `man opt_area`
`-no_declone`	Stops the software from decloning instances to reclaim area. By default, the software declones instances to reclaim area. *Data_type*: bool, optional
`-no_delete_buffer`	Prohibits Innovus from deleting buffers to reclaim area. *Default:* The software deletes buffers to reclaim area. *Data_type*: bool, optional
`-no_down_size`	Prohibits Innovus from downsizing instances to reclaim area. *Default:* The software downsizes instances to reclaim area. *Data_type*: bool, optional
`-hold_aware`	Runs the hold aware area reclaim. This is only supported after the design has been routed. *Data_type*: bool, optional

Example

By default area reduction is happening only on positive slack instances. To allow the tool to reclaim area on instances that have a negative slack, you can set a negative setup target slack:

```
set_db opt_setup_target_slack current_negative_slack
opt_area
```

Related Information

- `opt_design`
- opt Category Attributes chapter of *Innovus Stylus Common UI Text Command Reference*
- Optimizing Timing chapter of *Innovus Stylus Common UI User Guide*

opt_clock_skew

```
opt_clock_skew
[-help]
[-hold]
[-post_cts]
[-post_route]
```

Modifies the clock arrival time on sequential elements in order to improve the data path timing between two sequential elements. The `opt_clock_skew` command outputs depend on whether the clock tree is built or not.

- In pre-CTS mode, `opt_clock_skew` generates two files: a scheduling file and a latency file.
- In post-CTS and post-Route mode, `opt_clock_skew` inserts buffers and/or inverters in an existing clock tree. In addition, resizing of buffers and/or inverters can also be committed.

> Cadence recommends using the attribute, `opt_useful_skew`, and the command `opt_design -pre_cts`, `-post_cts`, or `post_route` to run useful skew optimization during timing optimization. For information on how to use `opt_design`, see "Optimizing Timing" in the *Innovus Stylus Common UI User Guide*.

Pre-CTS mode

The `opt_clock_skew` command identifies elements that can be advanced or delayed, and computes target delay for each. An advancable element is a sequential gate on which the clock signal can be safely made to arrive early without worsening the slack on the critical path. In general, this is true for sequential elements where the slack on the output pin (SQ) is negative (SQ < 0) and is worse than the slack on the input pin (SD) : (SQ < SD).

The `opt_clock_skew` command generates the following files:

- **Scheduling file**: The CTS scheduling file lists the delays to consider on sequential elements by the clock tree synthesis tool. When loaded in CTS, the scheduling file constrains CTS to build a tree that fulfills the targets computed by `opt_clock_skew`. The scheduling file must be loaded in addition to the original clock tree specification file (see "Examples" section below).

  The syntax of the scheduling file is as follows:

```
Macromodel pin lefPinName maxRiseDelay minRiseDelay maxFallDelay minFallDelay
inputCap
```

An entry in the configuration file is available to specify the scheduling file:

```
set rda_input ui_scheduling_file scheduling_file.cts
```

When the design is saved, the scheduling file is copied in the design directory, and the line above is added to the configuration file of the saved design.

- **Latency file**: This file lists the delays to be considered on sequential elements by the timing analysis tool. These delays are computed by `opt_clock_skew`.

  The file syntax is as follows:
  ```
  set_clock_latency rise delay pinName
  ```

  The timing engine takes the latency file into account when the following attribute is specified:
  ```
  set_db timing_analysis_useful_skew
  ```

  When building the timing graph, Innovus first loads the timing constraints, and then the latency file, as specified in the configuration file, for example
  ```
  set_rda_input (ui_latency_file) latency_file.sdc
  ```

  When the design is saved, the latency file is copied to the design directory, and the line above is added to the configuration file of the saved design.

Post-CTS and Post-Route Mode

The `opt_clock_skew` command identifies delayable elements and delays the clock for each element. A delayable element is a sequential element on which the clock signal can be safely delayed without worsening the slack on the critical path. In general, this is true for each sequential element where the slack on the data pin (SD) is negative (SD<0) and is worse than the slack on the output pin (SQ) : (SD < SQ). The `opt_clock_skew` command inserts and/or resizes buffers and/or inverters on an existing clock tree.

- The opt_clock_skew net name prefix is `FE_%sN`

- The opt_clock_skew instance name prefix is `FE_%sC`
- The default instance name added during useful skew optimization is `FE_USKC`
- The default net name added during useful skew optimization is `FE_USKN`

If SD > SQ, the elements before the D pin can be advanced, and `opt_clock_skew` can do resizing on the buffers and/or inverters to compensate the timing on Q pin.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `opt_clock_skew` parameter. For a detailed description of the command and all of its parameters, use the man command: `man opt_clock_skew`
`-hold`	Allows optimization for hold timing. *Data_type*: bool, optional
`-post_cts`	The `opt_clock_skew` command allows the use of an algorithm in the post_cts mode. When this parameter is specified, only setup timing optimization is performed. When it is specified along with the `-hold` parameter, only hold timing optimization is performed. This means, • The `opt_clock_skew -post_cts` command will only optimize setup timing. • The `opt_clock_skew -post_cts -hold` command will only optimize hold timing, without degrading setup timing. To optimize both setup and hold violations, run the following commands: `opt_clock_skew -post_cts` opt_clock_skew -post_cts -hold *Data_type*: bool, optional

`-post_route`	The `opt_clock_skew` command allows the use of an algorithm in the post_route mode. It forces the command to end with ECO routing for the modified clock nets. An `route_eco` is followed by clock skewing. The difference between post-CTS mode and post-Route mode is that the former is not limited to be legal only, while the latter is in a legal only mode.
	
	When this parameter is specified, only setup timing optimization is performed. When it is specified along with the `-hold` parameter, only hold timing optimization is performed.
	
	This means,
	• The `opt_clock_skew -post_route` command will only optimize setup timing.
	• The `opt_clock_skew -post_route -hold` command will only optimize hold timing, without degrading setup timing.
	
	To optimize both setup and hold violations, run the following commands:
	`opt_clock_skew -post_route`
	`opt_clock_skew -post_route -hold`
	
	Data_type: bool, optional

Examples

- The following example shows how to use `opt_clock_skew` in a pre-CTS flow without using `opt_design`:

  ```
  set_db timing_analysis_useful_skew true
  time_design -pre_cts
  set_scheduling_file scheduling_file.cts
  opt_clock_skew
  time_design -pre_cts
  ```

  Note: The recommended method for useful skew in pre-CTS mode is as follows:

  ```
  set_db opt_useful_skew true
  set_db opt_useful_skew_pre_cts true
  place_opt_design
  ```

- The following example shows how to use `opt_clock_skew` in post-CTS mode as an incremental fixing after normal post-CTS optimization:

  ```
  ccopt_design
  ```

```
opt_clock_skew -post_cts
time_design -post_cts
```

Related Information

- `place_opt_design`
- `set_scheduling_file`
- `time_design`
- The following chapters of *Innovus Stylus Common UI Text Command Reference*
  - opt Category Attributes
  - route Category Attributes
  - timing Category Attributes
- The following section in Optimizing Timing chapter of *Innovus Stylus Common UI User Guide*
  - Using Useful Skew

opt_design

```
opt_design
[-help]
[-drv]
[-exclude_nets fileName]
[-expanded_views]
[-hold]
[-hold_violation_report fileName]
[-ideal_clock]
[-include_nets fileName]
[-include_pins fileName]
[-incremental]
[-no_eco_route]
[-post_cts]
[-post_route]
[-pre_cts]
[-report_dir directoryName]
[-report_prefix fileNamePrefix]
[-setup]
[-targeted]
[-timing_debug_report]
[-use_transition_files]
```

Performs timing optimization before or after the clock tree is built, or after routing and generates timing reports. You can optimize with or without useful skew.

In multi-mode, multi-corner mode, `opt_design` optimizes all analysis views concurrently.

Performs the following optimizations:

- Corrects design rule violations.
- Reduces total negative slack.
- During the initial pass, optimizes setup time by working on the design's worst paths until the slack cannot be further improved, then, if the worst slack is not on a register-to-register path, optimizes the register-to-register paths on the second pass.
- Corrects hold time violations (optional).
- Optimizes useful skew (optional).
- Reclaims area (optional).
- Optimizes leakage power (optional).

- Merges flops to multi-bit FFs, or split the multi-bit FFs (optional).

Uses some or all of the following techniques, depending on the design stage (that is, before or after CTS, or after routing) and the specified parameters:

- Adds buffers.
- Resizes gates.
- Restructures the netlist.
- Remaps logic.
- Swaps pins.
- Deletes buffers.
- Moves instances.
- Applies useful skew.
- Applies layer promotion and/or NDR assertions.
- Applies flip-flop merge and split.

Sets the minimum recommended and appropriate set of attributes in the following categories:

- `timing` category attributes
- `route_trial` category attributes
- `opt` category attributes
- `extract_rc` category attributes

Tip: You can set additional parameters by specifying attributes using the appropriate `set_db` *`attribute_name value`* command or before using `opt_design`.

Before running `opt_design`, complete the following steps:

- Generate the default and detailed extraction scale factors by using the following commands:
  - `report_rc_factors`
- Set input transitions for the high fanout nets for delay calculation to use on high fanout networks.

Note: The `opt_remove_redundant_insts` attribute is allowed to remove unnecessary or redundant level shifter/isolation instances although those would be internally set as `dont_touch` by `opt_design`.

Parameters

Note: The "opt_design Parameter Matrix" section shows which parameters can be used together.

`-help`	Outputs a brief description that includes type and default information for each `opt_design` parameter. For a detailed description of the command and all of its parameters, use the man command: `man opt_design`
`-drv`	Corrects `max_cap` and `max_tran` violations. To correct `fanout_load` violations, specify `opt_fix_fanout_load` attribute before you specify `opt_design`. You cannot use this attribute if you specify `-incremental`. When applying this attribute, `opt_design` will fix only the DRV and not the setup/hold timing violation. *Data_type*: bool, optional
`-exclude_nets` *fileName*	
	Specifies the file that contains the hierarchical names of nets. Timing optimization excludes nets specified in the `-exclude_nets` file. If a net appears in both `-exclude_nets` and `-include_nets`, it is excluded. Note: Excluded nets, especially those with large fanout, might still be buffered to fix Setup timing problems. Use the dont_touch option in the SDC constraints file to prevent opt_design from modifying selected nets. For Hold Optimization however, the nets specified will be excluded when optimizing the Hold Timing. *Data_type*: string, optional
`-expanded_views`	Generates detailed view-specific timing reports at the end of the opt_design command run. A separate directory is created for each view. For example, if a design has two analysis views, view1 and view2, then the output reports are generated in the `./timingReports/view1` and `./timingReports/view2` directories. If the views were pruned during optimization, they will be restored at the end by this parameter and a view-specific summary and data will be reported on all the active views in the design. *Data_type*: bool, optional

`-hold`	Corrects hold violations. You cannot use this parameter if you specify `-pre_cts`. You can specify `-include_nets` or `-include_pins` if you use this parameter. This parameter creates a summary file in the output directory for the design. The file is named prefix_hold.summary. Use the -report_prefix parameter to specify the prefix. To report detailed debugging information, use `set_db opt_fix_hold_verbose true`. For more information, see opt Category Attributes. **Note**: Even though the hold-timing violation fixing algorithm is setup-timing aware, some setup timing degradation may appear due to different trialRoute topology during the post-CTS stage. In most cases, this can be overlooked because the violations will be easily fixed during the `opt_design -post_route` flow step. **Note**: You need to use the `add_fillers_cells` attribute to identify the filler cells. The `opt_design` command will remove filler cells that have been identified with the `add_fillers_cells` attribute at the beginning of optimization and add them back at the end of optimization. *Data_type*: bool, optional
`-hold_violation_report` *`fileName`*	Generates the following report files for the violation data that remains after hold fixing: *`fileName.txt`:* provides a list of the top 50 hold violation paths (text format) *`fileName.csv`*: provides a list of the top 50 hold violation paths (csv format) *`fileNameDetailed.txt`*: provides detailed information for the top 50 hold violation paths This parameter is used with the `-hold` parameter. **Example**: `opt_design -post_route -hold -hold_violation_report` *`fileName`* *Data_type:* string, optional

`-ideal_clock`	Performs timing optimization in post-CTS or post-Route modes using ideal clocks instead of propagated clocks. Using ideal clocks enables you to optimize timing without building a clock tree first, or ignore the existing clock tree, to close ideal timing. If this parameter is not specified, the software assumes the clock tree has been built, and optimizes timing based on propagated clocks. *Data_type*: bool, optional
`-include_nets` *fileName*	
	Specifies the file that contains the hierarchical names of nets to consider for design-rule violation repair or hold-timing violations fixing. Timing optimization excludes all other nets. If a net appears in both -exclude_nets and -include_nets, it is excluded. This option is only accepted when either `-drv` or `-hold` is selected. **Note**: The `-include_nets` parameter forces `opt_design` to work on all the selected nets, including the ones set using dont_touch. *Data_type*: string, optional
`-include_pins` *fileName*	
	Specifies the file that contains the hierarchical names of instance terminals to consider for timing optimization. Only nets connected to those terminals are optimized; for example, during hold fixing, delay cells are added only on the nets driving the terminals specified in the file. *Data_type*: string, optional

`-incremental`	Performs setup optimization incrementally on the violated paths until they cannot be optimized any more. Both, WNS and TNS are optimized. Both, high-effort path and low-effort path groups are optimized. Area reclaim may also take place if `-post_route` is not specified. A setup recovery is called depending on the timing bump after the re-route. This parameter has the following limitations: • You cannot use it if you also specify `-drv` or `-hold`. • You must have already run `opt_design` on all path groups before running it with `-incremental`. • When specified, this option disables power optimizations. *Data_type*: bool, optional
`-no_eco_route`	Prohibits the software from performing ECO routing with the NanoRoute® router during post-Route optimization. *Data_type*: bool, optional
`-post_cts`	Performs timing optimization on a design whose clock tree has been created. By default, -post_cts repairs design rule violations and setup violations. If the worst negative slack does not occur on a register-to-register path, `-post_cts` performs an additional optimization pass on the register-to-register critical paths when clock domains are not set. Does not repair maximum fanout design rule violations unless you first specify `opt_fix_fanout_load` attribute. *Data_type*: bool, optional

`-post_route`	Performs timing optimization on a design whose routing is complete. By default, `-post_route` repairs design rule violations, glitch Violations and setup violations on Base & SI Delay. If the worst negative slack does not occur on a register-to-register path, `-post_route` performs an additional optimization pass on the register-to-register critical paths when clock domains are not set. Maximum fanout design rule violations are not repaired by default. To do this, first specify `opt_fix_fanout_load` attribute. The `opt_design -post_route` command provides accurate results when Innovus runs using the footprintless flow methodology. By default, Innovus creates an internal footprint structure based on cell functionality. This methodology does not require you to specify a footprint file. For library database (LDB) conversion, this is the recommended methodology since all of it is automated. For more information, see the "Using the Footprintless Flow" section in the Optimizing Timing chapter of *Innovus Stylus Common UI User Guide*. *Data_type*: bool, optional
`-pre_cts`	Performs timing optimization on the placed design, before the clock tree is built. By default, `-pre_cts` repairs design rule violations and setup violations. If the worst negative slack does not occur on a register-to-register path, `-pre_cts` performs an additional optimization pass on the register-to-register critical paths when clock domains are not set. Maximum fanout design rule violations are not repaired by default. To do this, first specify `opt_fix_fanout_load` attribute. **Note**: Before optimizing a design in pre-CTS mode, you must break all timing loops by disabling arcs in the constraint file. If you do not disable the arcs, the software cannot make a valid comparison of WNS between two different runs since it might not break the loops at the same point each time. To disable the arcs, use the `set_disable_timing` command. *Data_type*: bool, optional
`-report_dir` *directoryName*	

	Specifies the directory where the software writes timing reports generated when this command runs. *Default*: `timingReports` *Data_type*: string, optional
`-report_prefix` *fileNamePrefix*	
	Specifies a prefix for `opt_design` report file names. *Default:* *DesignName_DesignStage*, where *DesignStage* is `pre-CTS`, `post-CTS`, or `post-Route`. *Data_type*: string, optional
`-setup`	Corrects setup violations for the post-Route optimization flow. When specified along with the `-hold` parameter, a combined setup and hold optimization is performed. For example, `opt_design -post_route -setup -hold` **Note**: This parameter is only applicable to the post-Route optimization flow and should be used along with the `-hold` parameter. *Data_type*: bool, optional

`-targeted`	This parameter is used only with the `-post_route` parameter. Specifying this parameter invokes target-based optimization (TBOpt), which is a fine grain local optimization functionality. This will target critical nodes at the post-Route stage and optimize these by giving worst negative slack (WNS), area, power, and max transition improvements with minimum disturbance to NanoRoute. This parameter can be used optionally along with the `opt_target_based_opt_file` and `opt_target_based_opt_file_only` attributes. **Note**: Area, power, and max transition targeting can only be done using the above settings. For more information on the syntax of a sample TBOpt file, see the opt Category Attributes documentation. Target-based optimization is best used after post-Route optimization is already completed and there are still some violating paths or nodes that might need further improvement. *Data_type*: bool, optional
`-timing_debug_report`	When specified, generates a machine-readable timing debug report after completing optimization. This report is used for debugging timing results using the timing debug feature in Stylus. **Note**: You can select the number of paths that should be reported per path group in the final summary reports by specifying the `opt_time_design_num_paths` attribute. The default number of paths is 50. *Data_type*: bool, optional
`-use_transition_files`	Uses the transition file specified by the `read_transition_file` command for transition violation fixing during post-Route SI optimization. *Data_type*: bool, optional

opt_design Parameter Matrix

The table below shows the compatibility of `opt_design` parameters.

opt_design	-pre_cts	-post_cts	-post_route	-drv	-setup	-hold	-ideal_clock	-incr	-no_eco_route
-pre_cts	NA	-	-	+	-	-	-	+	-
-post_cts	-	NA	-	+	-	+	+	+	+
-post_route	-	-	NA	+	+	+	+	+	+
-drv	+	+	+	NA	-	-	+	-	+
-setup	-	-	+	-	NA	+	+	-	+
-hold	-	+	+	-	+	NA	+	-	+
-ideal_clock	-	+	+	+	+	+	NA	+	+
-incr	+	+	+	-	-	-	+	NA	+
-no_eco_route	-	+	+	+	+	+	+	+	NA

Legend

+	You can use this parameter in the row with the parameter in the column.
-	You cannot use this parameter in the row with the parameter in the column.
NA	The parameters in the row and column are identical.

Examples

Pre-CTS Optimization

- To optimize timing for first time, use the following command:
  ```
  opt_design -pre_cts
  ```
- To repair design rule violations only, use the following command:
  ```
  opt_design -pre_cts -drv
  ```

- To run incremental setup-only optimization, use the following command:

  ```
  opt_design -pre_cts -incremental
  ```

- To optimize using low-effort mode for design prototyping, use the following attribute and command:

  ```
  set_db opt_effort low
  opt_design -pre_cts
  ```

  The `opt_design` command performs gate resizing and global buffer insertion, but does not perform netlist restructuring or design rule violation repair.

- To run optimization with useful skew, use the following attribute and command:

  ```
  set_db opt_useful_skew true
  set_db opt_useful_skew_pre_cts true
  opt_design -pre_cts
  ```

- To run optimization on specific path groups, use the following commands:

  ```
  group_path -name path_group_name -from from_list -to to_list -through through_list
  set_path_group_options  ..
  opt_design -pre_cts
  ```

  Or you can use:

  ```
  create_basic_path_groups
  set_path_group_options ..
  opt_design -pre_cts
  ```

  For example, to run optimization on reg2reg & reg2gate paths only, use the following commands:

  ```
  create_basic_path_groups -expanded
  set_path_group_options reg2out -target_slack -1000.0
  set_path_group_options in2reg -target_slack -1000.0
  set_path_group_options in2out -target_slack -1000.0
  opt_design -pre_cts
  ```

  Note: If you specify a path group, `opt_design` preserves the path group setting. In the previous example, subsequent commands may affect the reg2reg & reg2cgate path groups only. To set new groups, specify `reset_path_group -all` , followed by the definition of the groups you want to use.

For example, to reset the path group to a specific group of paths, use the following commands after you run `opt_design`:

```
reset_path_group -all
group_path -name pathgroup1 -through an1/Z -through an2/Z -through an3/Z
```

To specify all path groups, use the following command:

```
reset_path_group -all
```

- To reclaim area during optimization, use the following attribute and command:

  ```
  set_db opt_area_recovery true
  opt_design -pre_cts
  ```

- To run incremental optimization with useful skew, use the following attribute and command:

  ```
  set_db opt_useful_skew true
  opt_design -pre_cts -incremental
  ```

- To run an incremental optimization including reclaiming area, use the following attributes/commands:

  ```
  opt_design -pre_cts
  set_db opt_area_recovery true
  opt_design -pre_cts -incremental
  ```

Post-CTS Optimization

- To optimize timing after the clock tree is built, use the following command:

  ```
  opt_design -post_cts
  ```

  The `opt_design` command corrects design rule violations, then setup violations.

  Note: The software does not correct fanout violations in `-post_cts` and `-post_route` modes. To repair fanout violations, specify the following attribute before you run `opt_design`:

  ```
  set_db opt_fix_fanout_load true
  ```

- To repair setup and hold violations, use the following commands:

  ```
  opt_design -post_cts
  opt_design -post_cts -hold
  ```

- To repair design rule violations only, use the following command:

```
opt_design -post_cts -drv
```

- To repair hold violations only, use the following command:

  ```
  opt_design -post_cts -hold
  ```

- To run optimization on specific path groups, use the following commands:

  ```
  group_path -name path_group_name -from from_list -to to_list -through through_list
  set_path_group_options  ..
  opt_design -post_cts
  ```

  Or you can use:

  ```
  create_basic_path_groups
  set_path_group_options ..
  opt_design -post_cts
  ```

  For example, to run optimization on reg2reg & reg2gate paths only, use the following commands:

  ```
  create_basic_path_groups -expanded
  set_path_group_options reg2out -target_slack -1000.0
  set_path_group_options in2reg -target_slack -1000.0
  set_path_group_options in2out -target_slack -1000.0
  opt_design -post_cts
  ```

- To reclaim area during optimization, use the following attribute and command:

  ```
  set_db opt_area_recovery true
  opt_design -post_cts
  ```

- To take advantage of useful skew when optimizing timing in post-CTS mode, use the following attribute and command:

  ```
  set_db opt_useful_skew true
  opt_design -post_cts
  ```

  If you specify `-no_eco_route`, the software uses trial routing to estimate clock delays.

- To run post-CTS optimization if your design has a clock mesh, use the following attribute and command:

  ```
  set_db opt_useful_skew false
  opt_design -post_cts
  ```

- To run incremental optimization on specific path groups, use the following commands:

  ```
  group_path -name path_group_name -from from_list -to to_list -through through_list
  set_path_group_options  ..
  opt_design -post_cts -incremental
  ```

  For example, to run incremental optimization on reg2reg & reg2gate paths only, use the following commands:

  ```
  create_basic_path_groups -expanded
  set_path_group_options reg2out -target_slack -1000.0
  set_path_group_options in2reg -target_slack -1000.0
  set_path_group_options in2out -target_slack -1000.0
  opt_design -post_cts -incremental
  ```

- To optimize setup time incrementally and reduce area, use the following attribute and command:

  ```
  set_db opt_area_recovery true
  opt_design -post_cts -incremental
  ```

- To take advantage of useful skew when optimizing timing in incremental post-CTS mode, use the following attribute and command:

  ```
  set_db opt_useful_skew true
  opt_design -post_cts -incremental
  ```

- To run incremental post-CTS optimization if your design has a clock mesh, use the following attribute and command:

  ```
  set_db opt_useful_skew false
  opt_design -post_cts -incremental
  ```

Post-Route Optimization

- To optimize base & SI timing after detailed routing (repair setup violations), use the following command:

  ```
  opt_design -post_route
  ```

- If you wish to optimize only the base timing after detailed routing, use the following attribute and command:

  ```
  set_db delaycal_enable_si false
  opt_design -post_route
  ```

- **Note:** The software does not correct fanout violations in `-post_cts` and `-post_route` modes. To repair fanout violations, set the following attribute before you run `opt_design`:
  ```
  set_db opt_fix_fanout_load true
  ```
- To correct hold base & SI timing violations, specify the following command:
  ```
  opt_design -post_route -hold
  ```
- To correct setup and hold (both base & SI) violations, use the following commands:
  ```
  opt_design -post_route
  opt_design -post_route -hold
  ```

  The software repairs a hold violation only if it does not make setup slack worse than the setup target slack on a path. The software repairs a hold violation only if it does not make setup slack worse than the setup target slack on a path. An alternative way is to run setup and hold fixing within one command, and this may reduce the number of ecoRoute(s):
  ```
  opt_design -post_route -setup -hold
  ```
- To take clock reconvergence pessimism removal (CPPR) into consideration when running timing optimization, use the following `timing category` commands and attributes before you run `opt_design`. For example:
  ```
  set_timing_derate -max -clock -early 0.8 -late 1.2
  set_timing_derate -min -clock -early 0.8 -late 1.2
  set_db timing_analysis_cppr true
  opt_design -post_route
  ```

Related Information

- `create_basic_path_groups`
- `group_path`
- `set_db`
- `set_path_group_options`
- `set_timing_derate`
- `time_design`
- `write_ilm`
- The following chapters of *Innovus Stylus Common UI Text Command Reference*
  - timing Category Attributes
  - delaycal Category Attributes
  - route Category Attributes
  - opt Category Attributes
  - si Category Attributes
  - route_early_global Category Attributes
- Optimizing Timing chapter in the *Innovus Stylus Common UI User Guide*

opt_power

```
opt_power
[-help]
[-allow_resize]
[-effort_level {high}]
[-force]
[-pre_cts | -post_cts | -post_route]
[-pre_place]
[-seq_only]
```

Optimizes the power consumption of the design by swapping gates for those with lower power, or deleting buffers/inverters without degrading timing. This works in conjunction with the `opt_leakage_to_dynamic_ratio` attribute to determine the type of optimization being performed - leakage, dynamic, or a combination of both.

Since the default for opt_leakage_to_dynamic_ratio attribute is 1.0, opt_power will perform leakage optimization only by default using LEF-safe optimization (if no other switch is set). It is vital to set up the design correctly before starting power optimization. For more information, see the Optimizing Power During opt_design section of the "Optimizing Timing" chapter in the *Innovus Stylus Common UI User Guide*.

The setup steps include the following:

- Setting the correct power views for both leakage and dynamic optimization
- Ensuring that the libraries contain `cell_leakage_power` information
- Ensuring that the switching activity has been correctly specified either through `read_activity_file -format {VCD | TCF | SAF} file_name` or the `set_default_switching_activity` command

Parameters

`-help`	Outputs a brief description that includes type and default information for each `opt_power` parameter. For a detailed description of the command and all of its parameters, use the man command: `man opt_power`.

`-allow_resize`	Specifies that optimization uses resizing for power reclaim and, if needed, for recovery. Since the cells will be of different physical footprint, placement refinement and routing ECO will be used. The DRV and timing recovery steps will also be performed, as required. Due to these extra steps, an increase in runtime is expected. As this is a more aggressive optimization, the power savings should be superior to the default opt_power values, but this may cause timing degradations in some cases. *Data_type*: bool, optional
`-effort_level {high}`	Specifies the effort level as `high`. When this parameter is specified, power optimization uses aggressive reclaim. The software performs a timing recovery step and there is a reasonably large increase in runtime on an average as compared to when regular power optimization is performed. *Data_type*: enum, optional
`-force`	Resizes each instance to a cell with the lowest leakage power, disregarding design rule violations and optimum timing. This only works when `opt_leakage_to_dynamic_ratio` attribute is set to `1.0`. *Data_type*: bool, optional
`-pre_cts`	Performs power optimization on the placed design, before the clock trees are created. *Data_type*: bool, optional
`-pre_place`	When specified, performs power optimization on an unplaced design. This parameter must be used with the `-force` parameter. *Data_type*: bool, optional
`-post_cts`	Performs power optimization on a design whose clock trees have been created. *Data_type*: bool, optional
`-post_route`	Performs power optimization on a design whose routing is complete. *Data_type*: bool, optional
`-seq_only`	Specifies that dynamic power opt should be done on sequential cells only. This parameter works with the `-force` parameter in all modes; `pre_cts`, `pre_place`, `post_cts`, and `post_route`. *Data_type*: bool, optional

Examples

- Use the following attribute and command to optimize the design targeting total power at the pre-CTS stage:

```
set_db opt_leakage_to_dynamic_ratio 0.5
opt_power -pre_cts
```

- Use the following command to optimize the design, targeting leakage power only at the post-CTS stage (assuming the `opt_leakage_to_dynamic_ratio` attribute has been untouched or has been set to the default of 1.0):

```
opt_power -post_cts
```

- Use the following commands to optimize dynamic power only at the post-Route stage:

```
set_db opt_leakage_to_dynamic_ratio 0.0
opt_power -post_route
```

- Use the following attribute and command to perform resizing for total power optimization in the design at the pre-CTS stage:

```
set_db opt_leakage_to_dynamic_ratio 0.5
opt_power -pre_cts -allow_resize
```

- Use the following attribute and command to perform aggressive dynamic power optimization in the design at the post-CTS stage:

```
set_db opt_leakage_to_dynamic_ratio 0.0
opt_power -post_cts -effort_level high
```

- Use the following attribute and command to optimize leakage power by forcing all cells to their lowest leakage equivalent, disregarding DRV and timing degradations at the pre-CTS stage:

```
set_db opt_leakage_to_dynamic_ratio 1.0
opt_power -pre_cts -force
```

- Use the following attribute and command to optimize dynamic power by swapping sequential cells only in the design at the pre-CTS stage:

```
set_db opt_leakage_to_dynamic_ratio 0.0
opt_power -pre_cts -seq_only
```

- Use the following attribute and command to allow the tool to swap gates, including the gates with negative slack, up to the current negative slack at the post-Route stage:

```
set_db opt_setup_target_slack <current_negative_slack>
opt_power -post_route
```

- To prevent power optimization from working on near critical paths, you can set a different target to that of the timing optimization. A typical example is shown below where power optimization will not work on paths less than 0.1ns but the target for timing will remain `0.0`:

```
set_db opt_setup_target_slack 0.0
set_db opt_area_recovery_setup_target_slack 0.1
set_db opt_leakage_to_dynamic_ratio 0.5
opt_power -pre_cts
```

Related Information

- opt Category Attributes chapter of *Innovus Stylus Common UI Text Command Reference*
- `read_activity_file`
- `set_default_switching_activity`
- The following section in Optimizing Timing chapter of *Innovus Stylus Common UI User Guide*
  - Optimizing Power During opt_design

opt_signoff

```
opt_signoff
[-help ]
[-all]
[-area]
[-drv]
[-dynamic]
[-hold]
[-leakage]
[-no_eco_route]
[-power]
[-report_dir dir_name]
[-report_full_clock_path]
[-report_prefix prefix_name]
[-setup]
```

Runs timing, area, DRV, dynamic or leakage optimization on signoff timing.

This command must be run on a design that has gone through the implementation flow. It allows you to fix timing violations or to improve area and power visible in signoff STA mode while staying in the implementation tool.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `opt_signoff` parameter. For a detailed description of the command and all of its parameters, use the man command: `man opt_signoff`
`-all`	Performs DRV, Setup, Hold, and Power optimization in the path-based analysis (PBA) mode. It also includes embedded calls to route_eco, parasitic extraction and signoff timing analysis. **Note**: The all-in-one mode does not replace all customized fine-tuned flows. It provides the Stylus users easy access to Tempus ECO technology. The existing Tempus ECO flow using individual optimization steps within Stylus is still supported and can sometimes be more efficient than the all-in-one mode. *Data_type*: bool, optional

`-area`	Reclaims area on the netlist without causing any DRV/Setup/Hold degradation. *Data_type*: bool, optional
`-drv`	Triggers DRV fixing in signoff timing mode. *Data_type*: bool, optional
`-dynamic`	Triggers dynamic optimization in signoff timing mode. *Data_type*: bool, optional
`-hold`	Triggers hold fixing in signoff timing mode. *Data_type*: bool, optional
`-leakage`	Triggers leakage optimization in signoff timing mode. *Data_type*: bool, optional
`-power`	Optimizes total power in the design. During total power optimization, the software performs swapping and sizing on both the combinational and the sequential cells, and also removes buffers. This reduces leakage, internal, and switching power in the design. *Data_type*: bool, optional
`-no_eco_route`	Allows `opt_signoff` command to skip ECO routing and final signoff STA. *Data_type*: bool, optional
`-report_dir`	Specifies the directory in which the reports will be saved. **Note**: In case `opt_signoff` is reusing ECO timing DB generated earlier by `time_design_signoff` or when the `-no_eco_route` parameter is applied, no timing reports are saved. *Data_type*: string, optional
`-report_full_clock_path`	When this parameter is specified, the detailed timing report that is generated contains the full clock path. *Data_type*: bool, optional
`-report_prefix`	Specifies the prefix for the timing report file names. *Data_type*: string, optional

`-setup`	Triggers setup fixing in signoff timing mode. *Data_type*: bool, optional

Examples

- Use the following commands to perform setup optimization followed by a leakage power optimization. The setup optimization is with no ECO route, while the leakage power optimization calls the ECO Route along with final signoff STA results.

  ```
  opt_signoff -setup -no_eco_route
  opt_signoff -leakage
  ```

- Use the following commands to perform an area reclaim optimization that is based on external spef files:

  ```
  read_spef -rc_corner rc_max rc_max.spef.gz
  read_spef -rc_corner rc_min rc_min.spef.gz
  time_design_signoff -report_only
  opt_signoff -area
  ```

- Use the following command to dump out the final STA results in a folder named "RPT". The timing report files inside the "`RPT`" folder will be named "`SIGNOFF_ECO_*`".

  ```
  opt_signoff -hold -report_dir RPT -report_prefix SIGNOFF_ECO
  ```

- Use the following set of commands to perform DRV, Setup, Hold, and Power optimization in the PBA mode:

  ```
  set_multi_cpu_sage -local_cpu 16 -remote_host 4 -cpu_per_remote_host 4
  set_db opt_signoff_allow_skewing true
  set_db opt_signoff_clock_cell_list {…}
  opt_signoff -all
  ```

  Stylus will run Tempus ECO for PBA-based DRV, timing, and power closure. The Setup fixing will be done using useful skew technology. Signoff timing analysis will use 4 remote hosts each using 4 CPUs while ECO fixing, ECO routing, and parasitic extraction will use 16 CPUs.

Related Information

- opt_signoff Category Attributes chapter of *Innovus Stylus Common UI Text Command Reference*
- `read_spef`
- `time_design_signoff`
- The following section in Optimizing Timing chapter of *Innovus Stylus Common UI User Guide*
  - Using Signoff Timing Analysis to Optimize Timing and Power

opt_signoff Category Attributes

The root attributes in the `opt_signoff` category are:

- opt_signoff_add_inst
- opt_signoff_add_load
- opt_signoff_allow_skewing
- opt_signoff_along_route_buffering
- opt_signoff_buffer_cell_list
- opt_signoff_check_drv_from_hold_views
- opt_signoff_check_type
- opt_signoff_clock_cell_list
- opt_signoff_clock_max_level
- opt_signoff_delete_inst
- opt_signoff_disable_geometry_checks
- opt_signoff_drv_margin
- opt_signoff_eco_file_prefix
- opt_signoff_em_type
- opt_signoff_fix_cell_em
- opt_signoff_fix_clock_drv
- opt_signoff_fix_data_drv
- opt_signoff_fix_glitch
- opt_signoff_fix_hold_allow_setup_optimization
- opt_signoff_fix_hold_allow_setup_tns_degrade
- opt_signoff_fix_hold_with_margin
- opt_signoff_fix_ir_drop
- opt_signoff_fix_max_cap
- opt_signoff_fix_osus_glitch

- opt_signoff_fix_max_transition
- opt_signoff_fix_slack_in_range_lower_limit
- opt_signoff_fix_slack_in_range_upper_limit
- opt_signoff_fix_si_slew
- opt_signoff_fix_xtalk
- opt_signoff_hold_target_slack
- opt_signoff_hold_xtalk_delta_threshold
- opt_signoff_hold_xtalk_slack_threshold
- opt_signoff_ignore_drv_checks
- opt_signoff_keep_tmp_files
- opt_signoff_legal_only
- opt_signoff_load_cell_list
- opt_signoff_max_cap_margin
- opt_signoff_max_transition_margin
- opt_signoff_max_opt_runtime
- opt_signoff_max_paths
- opt_signoff_max_runtime
- opt_signoff_max_slack
- opt_signoff_num_report_paths
- opt_signoff_nworst
- opt_signoff_optimize_core_only
- opt_signoff_optimize_replicated_modules
- opt_signoff_optimize_sequential_cells
- opt_signoff_partition_list_file
- opt_signoff_pba_eco_route
- opt_signoff_pba_effort
- opt_signoff_post_mask

- opt_signoff_post_mask_decap_cells
- opt_signoff_post_mask_filler_cells
- opt_signoff_post_mask_gate_array_filler_cells
- opt_signoff_post_mask_logic_cells
- opt_signoff_post_sta_tcl
- opt_signoff_power_aware
- opt_signoff_power_opt_focus
- opt_signoff_pre_sta_tcl
- opt_signoff_prefix
- opt_signoff_preserve_filler
- opt_signoff_read_eco_opt_db
- opt_signoff_read_irdrop_db
- opt_signoff_resize_inst
- opt_signoff_retime
- opt_signoff_retime_mode
- opt_signoff_retime_depth
- opt_signoff_retime_depth_early
- opt_signoff_routing_congestion_aware
- opt_signoff_select_hold_endpoints
- opt_signoff_select_drv_net_file
- opt_signoff_select_drv_pin_file
- opt_signoff_select_setup_endpoints
- opt_signoff_setup_recovery
- opt_signoff_setup_target_slack
- opt_signoff_setup_xtalk_delta_threshold
- opt_signoff_setup_xtalk_slack_threshold
- opt_signoff_skip_drv_net_file

- opt_signoff_set_hold_endpoints_margin
- opt_signoff_set_setup_endpoints_margin
- opt_signoff_swap_inst
- opt_signoff_verbose
- opt_signoff_write_eco_opt_db

To set the value of any root attribute, use the `set_db` command:

```
set_db attribute_name value
```

To get the current value of any root attribute, use the `get_db` command:

```
get_db attribute_name
```

To get the current value of all attributes in the `opt_signoff` category, use the following command:

```
get_db -category opt_signoff
```

opt_signoff_add_inst

```
opt_signoff_add_inst {true | false}
```

Default: `true`

Data_type: bool, read/write

Specifies whether or not Setup/Hold/DRV optimizer is allowed to add buffer/delay/swap cell instances.

Note: If set, this attribute must be applied before opt_signoff.

Related Commands

- `opt_signoff`
- `time_design_signoff`

opt_signoff_add_load

```
opt_signoff_add_load {true | false}
```

Default: `false`

Data_type: bool, read/write

Specifies whether Hold optimizer is allowed to add dummy cell instances.

Note: If set, this attribute must be applied before `opt_signoff`.

Related Commands

- `opt_signoff`
- `time_design_signoff`

opt_signoff_allow_skewing

```
opt_signoff_allow_skewing {true | false}
```

Default: `false`

Data_type: bool, read/write

Allows clock skewing to be done during Setup timing optimization to further improve Setup slack. When set to true, `opt_signoff -setup` works on the last level of the clock tree to modify the clock skew in order to improve Setup timing closure. Modification on the clock tree can be either resizing of the last level of the clock tree or buffering to pad endpoint clock pin.

Default: false

Note:

- If set, this parameter must be applied before `opt_signoff`.
- When `opt_signoff_allow_skewing` is active, `opt_signoff` will automatically unfix all clock tree instances/nets. This is also performed when implementing the ECOs.
- As for data path optimization, skewing is not allowed to create new Hold violated paths.

Example:

The following commands perform clock skewing during setup fixing. Only `CKBUFX1` and `CKBUFX4` can be added to the clock tree.

```
set_db opt_signoff_clock_cell_list {CKBUFX1 CKBUFX4}
set_db opt_signoff_allow_skewing true
opt_signoff -setup
```

Related Commands

- opt_signoff
- time_design_signoff

opt_signoff_along_route_buffering

```
opt_signoff_along_route_buffering {true | false}
```

Default: false

Data_type: bool, read/write

When set to 1, the software uses node location provided through the parasitic data in order to insert buffers along the routing topology not only at the start or end of the net. This helps to better balance the slew or the load for long nets, which is typically needed for setup timing optimization or DRV fixing.

Note:

- A prerequisite of this parameter is that parasitic with the node location is provided to signoffOptDesign. In addition, the node location parasitic should be read using the -keep_star_node_locations parameter. For example: read_spef -keep_star_node_locations.
- If set, this attribute must be applied before opt_signoff.

Related Commands

- opt_signoff
- time_design_signoff

opt_signoff_buffer_cell_list

```
opt_signoff_buffer_cell_list cell_list
```

Default: "" (empty string)

Data_type: string, read/write

Specifies a list of buffer/delay cells that can be used for the timing optimizer, regardless of whether those cells have a dont_use attribute. When the list is empty, the software will automatically identify

the list of usable buffer/delay cells.

Note: If set, this attribute must be applied before `opt_signoff`.

Related Commands

- `opt_signoff`
- `time_design_signoff`

opt_signoff_check_drv_from_hold_views

opt_signoff_check_drv_from_hold_views `{true | false}`

Default: `false`

Data_type: bool, read/write

Enables optimization of DRV violations (max_capacitance/max_transition) in hold views as well in setup views during DRV optimization. Also, when enabled, this attribute ensures that timing/area/power optimization will not create any new DRV violation in hold and setup views.

Note: If set, this attribute must be applied before `opt_signoff` and `time_design_signoff`.

Related Commands

- `opt_signoff`
- `time_design_signoff`

opt_signoff_check_type

```
opt_signoff_check_type {early | late | both}
```

Default: both

Data_type: enum, read/write

Specifies the type of ECO Timing DB to be generated in PBA mode. The `early` argument specifies hold ECO Timing DB and `late` specifies setup ECO Timing DB. When set to `both`, the ECO Timing DB generation will be done in PBA for both Setup and Hold timing.

Note:

- This attribute is only active when the `opt_signoff_retime` attribute is being set.
- If set, this attribute must be applied before `time_design_signoff`.

Related Commands

- `opt_signoff`
- `time_design_signoff`

opt_signoff_clock_cell_list

`opt_signoff_clock_cell_list` *`cell_list`*

Default: "" (empty string)

Data_type: string, read/write

Restricts the list of buffers that are allowed to be used by ECO on the clock tree.

Note: If set, this attribute must be applied before `opt_signoff`.

Related Commands

- `opt_signoff`
- `time_design_signoff`

opt_signoff_clock_max_level

`opt_signoff_clock_max_level` *`integer`*

Default: `0`

Data_type: int, read/write

Specifies the maximum levels in clock skewing thus limiting the amount of clock tree levels and avoiding the creation of an unbalanced clock tree. This attribute sets an upper boundary when adding a buffer or pair of inverters to the clock nets. The software counts the number of levels in the clock tree branch that is being modified to ensure that the maximum level will be equal to or less than the value set by this attribute.

Note: If set, this attribute must be applied before `opt_signoff`

Related Commands

- opt_signoff
- time_design_signoff

opt_signoff_delete_inst

```
opt_signoff_delete_inst {true | false}
```

Default: true

Data_type: bool, read/write

Enables buffer deletion when doing timing, power, or area reduction. This attribute is not honored by the `-leakage` parameter of `opt_signoff` since it performs only swapping.

Note: If set, this attribute must be applied before `opt_signoff`.

Related Commands

- opt_signoff
- time_design_signoff

opt_signoff_disable_geometry_checks

```
opt_signoff_disable_geometry_checks {true | false}
```

Default: false

Data_type: bool, read/write

Performs Vth swapping without checking the pins and geometry of a design. You can relax/stop/avoid the geometry checks during Vth swapping so that the tool just checks the cell size while performing Vth swapping. This will lead to more Vth swaps. The counterpart to this is that ecoRouting will be needed once the ECOs are applied since pin geometries might always match between old and new cells.

Note: If set, this attribute must be applied before `opt_signoff`.

Related Commands

- opt_signoff
- time_design_signoff

opt_signoff_drv_margin

```
opt_signoff_drv_margin value
```

Default: 0

Data_type: double, read/write

Specifies the ratio by which DRV check would be relaxed or tightened. It scales the maximum capacitance and maximum transition constraints according to the margin decimal value specified. For example, specifying a value of 0.2 for this attribute tightens the margin (makes it more pessimistic) by 20 percent during optimization and specifying a value of -0.2 relaxes the margin (makes it more optimistic) by 20 percent.

This margin is honored for DRV optimization.

Note: The software only accepts values from -0.7 to +0.7. A positive value indicates tighter constraints.

Note: If set, this attribute must be applied before `signoffOptDesign`.

Related Commands

- opt_signoff
- time_design_signoff

opt_signoff_eco_file_prefix

```
opt_signoff_eco_file_prefix prefix
```

Default: "" (empty string)

Data_type: string, read/write

Specifies the prefix name for the ECO files that are generated.

Note: If set, this attribute must be applied before `opt_signoff`.

Related Commands

- opt_signoff
- time_design_signoff

opt_signoff_em_type

```
opt_signoff_em_type {all | avg | rms | peak}
```

Default: `all`

Data_type: enum, read/write

Specifies the current waveform calculation method when doing EM fixing.

Related Commands

- opt_signoff
- time_design_signoff

opt_signoff_fix_cell_em

```
opt_signoff_fix_cell_em {true | false}
```

Default: `false`

Data_type: bool, read/write

When set to `true`, fixes the cell-level signal electromigration violations during DRV fixing using resizing and additional transforming techniques.

Note:

- If set, this attribute must be applied before `opt_signoff -drv`.
- If set during DRV fixing, the software will fix only the cell EM violations. This means that fixing of max_transition/max_capacitance/SI glitch/SI Xtalk/IR drop will be disabled.

Related Commands

- opt_signoff

opt_signoff_fix_clock_drv

`opt_signoff_fix_clock_drv {true | false}`

Default: `false`

Data_type: bool, read/write

Fixes maxCap/maxTran violations on the clock networks. This is performed only when the `opt_signoff -drv` command is run.

Note: For clock network DRV fixing, only the cells listed in the clock cell list (specified using the `opt_signoff_clock_cell_list` attribute), is used. The DRV fixing is not performed if a cell list is not provided.

Note: If set, this attribute must be applied before `opt_signoff`.

opt_signoff_fix_data_drv

`opt_signoff_fix_data_drv {true | false}`

Default: `false`

Data_type: bool, read/write

Fixes maxCap/maxTran violations on the data networks. This is performed only when the `opt_signoff -drv` command is run.

opt_signoff_fix_glitch

`opt_signoff_fix_glitch {true | false}`

Default: `false`

Data_type: bool, read/write

Performs SI glitch fixing during DRV fixing using resizing and buffering techniques.

Notes:

- If set, this attribute must be applied before `time_design_signoff`.
- The glitch fixing is done before max_tran/max_cap fixing.

opt_signoff_fix_hold_allow_setup_optimization

```
opt_signoff_fix_hold_allow_setup_tns_degrade {true | false}
```

Default: `false`

Data_type: bool, read/write

When set to `true`, incremental Setup fixing is performed during Hold fixing in order to create more Setup margins on Hold violated nodes that could not be fixed earlier due to Setup timing conflict.

Note: If set, this attribute must be applied before `opt_signoff`.

opt_signoff_fix_hold_allow_setup_tns_degrade

```
opt_signoff_fix_hold_allow_setup_tns_degrade {true | false}
```

Default: `false`

Data_type: bool, read/write

When specified, the Setup Total Negative Slack can be degraded during Hold time violations fixing while still maintaining the Setup Worst Negative Slack in every Setup active views.

Note: If set, this attribute must be applied before `opt_signoff`.

Related Commands

- `opt_signoff`
- `time_design_signoff`

opt_signoff_fix_hold_with_margin

```
opt_signoff_fix_hold_with_margin value
```

Default: `0`

Data_type: double, read/write

Specifies a slack margin value in nanoseconds for Hold time violations fixing. The purpose is to overfix any Hold violations by the amount of the margin value. This margin is not applied on nets that are already meeting the Hold target slack.

Note: If set, this attribute must be applied before `opt_signoff`.

Related Commands

- opt_signoff
- time_design_signoff

opt_signoff_fix_ir_drop

```
opt_signoff_fix_ir_drop  {true | false}
```

Default: `false`

Data_type: bool, read/write

Fixes the IR drop voltages for all instances in the design.

Note: If set, this attribute must be applied before `opt_signoff`.

Related Commands

- opt_signoff
- time_design_signoff

opt_signoff_fix_max_cap

```
opt_signoff_fix_max_cap  {true | false}
```

Default: `true`

Data_type: bool, read/write

Specifies that nets with max_cap violations should be fixed.

Note: If set, this attribute must be applied before `opt_signoff`.

Related Commands

- opt_signoff
- time_design_signoff

opt_signoff_fix_osus_glitch

`opt_signoff_fix_osus_glitch {true | false}`

Default: `false`

Data_type: bool, read/write

When set to `true`, fixes nets with overshoot and undershoot glitches during DRV optimization.

Related Commands

- `opt_signoff`
- `time_design_signoff`

opt_signoff_fix_max_transition

`opt_signoff_fix_max_transition {true | false}`

Default: `true`

Data_type: bool, read/write

Specifies that nets with max_tran violations should be fixed.

Note: If set, this attribute must be applied before `opt_signoff`.

Related Commands

- `opt_signoff`
- `time_design_signoff`

opt_signoff_fix_slack_in_range_lower_limit

`opt_signoff_fix_slack_in_range_lower_limit` *`value_in_ns`*

Default: `-1e+30`

Data_type: bool, read/write

Sets the lower slack limit for a violated path to be considered during timing closure. The value is specified in ns.

Related Commands

- `opt_signoff`
- `time_design_signoff`

opt_signoff_fix_slack_in_range_upper_limit

`opt_signoff_fix_slack_in_range_upper_limit` *`value_in_ns`*

Default: `1e+30`

Data_type: bool, read/write

Sets the upper slack limit for a violated path to be considered during timing closure. The value is specified in ns.

Related Commands

- `opt_signoff`
- `time_design_signoff`

opt_signoff_fix_si_slew

`opt_signoff_fix_si_slew {true | false}`

Default: `false`

Data_type: bool, read/write

Reports signal integrity (SI) slews violations and fixed during DRV optimization. This is run as the first step of optimization before proceeding for any other incremental optimization.

When this attribute is set to `true`, it automatically sets the value of the `opt_signoff_max_tran` attribute to `true`.

Note: If set, this attribute must be applied before `opt_signoff` and `time_design_signoff`.

Related Commands

- opt_signoff
- time_design_signoff

opt_signoff_fix_xtalk

```
opt_signoff_fix_talk  {true | false}
```

Default: false

Data_type: bool, read/write

Reduces the crosstalk on the net that violates the thresholds set by you.

Note: If set, this attribute must be applied before `opt_signoff` and `time_design_signoff`.

Related Commands

- opt_signoff
- time_design_signoff

opt_signoff_hold_target_slack

```
opt_signoff_hold_target_slack value
```

Default: 0

Data_type: double, read/write

Specifies a target slack value in nanoseconds for hold timing optimize.

Note: If set, this attribute must be applied before `opt_signoff`.

Related Commands

- opt_signoff
- time_design_signoff

opt_signoff_hold_xtalk_delta_threshold

```
opt_signoff_hold_xtalk_delta_threshold value_in_ns
```

Default: `0.3 ns`

Data_type: double, read/write

Fixes the nets with the crosstalk delta delay value equal to or greater than a specified threshold value for hold views.

Note: If set, this attribute must be applied before `opt_signoff`.

Related Commands

- `opt_signoff`
- `time_design_signoff`

opt_signoff_hold_xtalk_slack_threshold

```
opt_signoff_hold_xtalk_slack_threshold value_in_ns
```

Default: `1000 ns`

Data_type: double, read/write

Fixes nets with the slack threshold value equal to or less than a specified threshold value for hold views.

Note: If set, this attribute must be applied before `opt_signoff`.

Related Commands

- `opt_signoff`
- `time_design_signoff`

opt_signoff_ignore_drv_checks

```
opt_signoff_ignore_drv_checks  {true | false}
```

Default: `false`

Data_type: bool, read/write

Specifies that DRV checks should be ignored in setup/hold/area/leakage/dynamic optimization.

Note: If set, this attribute must be applied before `opt_signoff`.

Related Commands

- `opt_signoff`
- `time_design_signoff`

opt_signoff_keep_tmp_files

```
opt_signoff_keep_tmp_files  {true | false}
```

Default: `false`

Data_type: bool, read/write

Keeps all the temporary files generated by the distributed multi-mode multi-corner (DMMMC) and Tempus clients for debugging purposes.

Note: If set, this attribute must be applied before `opt_signoff`.

Related Commands

- `opt_signoff`
- `time_design_signoff`

opt_signoff_legal_only

```
opt_signoff_legal_only {true | false}
```

Default: true

Data_type: bool, read/write

Enables the tool to use the physical data to search for legal location while doing any buffer insertion or instance movement. This ensures that there is no instance overlap at the end of the ECO fixing stage, and after sourcing the ECO file in the implementation tool, you can directly run ECO routing.

When this attribute is set to `false`, `opt_signoff` will not check for legal location although physical

data is provided. In this mode, `opt_signoff` behaves in a Logical mode flow, which means that physical data will be ignored and this might create an instance overlap and will not check local density while inserting buffers. So it is recommended that you first run ECO fixing in the legal-only mode (which is the default mode) and then, depending on the requirement, you may perform an incremental run by setting `opt_signoff_legal_only` to `false`.

Default: `true`

Note: If set, this attribute must be applied before `opt_signoff`.

Related Commands

- `opt_signoff`
- `time_design_signoff`

opt_signoff_load_cell_list

`opt_signoff_load_cell_list` *`cell_list`*

Default: `""` (empty string)

Data_type: string, read/write

Specifies the list of load cells to be inserted during Hold fixing, regardless of whether those cells have a dont_use attribute.

Note: If set, this attribute must be applied before `opt_signoff`.

Related Commands

- `opt_signoff`
- `time_design_signoff`

opt_signoff_max_cap_margin

`opt_signoff_max_cap_margin` *`cap_margin_value`*

Default: 0

Minimum value: -10.000000

Maximum value: 0.990000

Data_type: double, read/write

Specifies the ratio by which the max_cap check should be relaxed or tightened.

Note: If set, this attribute must be applied before `opt_signoff`.

Related Commands

- `opt_signoff`
- `time_design_signoff`

opt_signoff_max_transition_margin

`opt_signoff_max_transition_margin` *`tran_margin_value`*

Default: 0

Minimum value: -10.000000

Maximum value: 0.990000

Data_type: double, read/write

Specifies the ratio by which the max_tran check should be relaxed or tightened.

Note: If set, this attribute must be applied before `opt_signoff`.

Related Commands

- `opt_signoff`
- `time_design_signoff`

opt_signoff_max_opt_runtime

`opt_signoff_max_opt_runtime` *`integer`*

Default: 0

Data_type: int, read/write

Specifies the time out period for only the timing/power optimization phase after which

opt_signoff will exit the session.

Note: If set, this attribute must be applied before `opt_signoff`.

Related Commands

- `opt_signoff`
- `time_design_signoff`

opt_signoff_max_paths

```
opt_signoff_max_paths integer
```

Default: -1

Data_type: int, read/write

Specifies the worst path found for each endpoint up to the number specified by the -maxPaths parameter during ECO Timing DB generation. By default, the tool will automatically set a value for `opt_signoff_max_paths` that would lead to good PBA coverage of the design within reasonable run time. You can overwrite this by setting a specific `opt_signoff_max_paths` value, but it is recommended to keep it larger than 1 million. In any case, you must ensure that `opt_signoff_max_paths` value is superior or equal than the `opt_signoff_nworst` value multiplied by the number of GBA violated endpoints.

Note:

- This attribute is only active when the `opt_signoff_retime` attribute is set.
- If set, this attribute must be applied before `time_design_signoff`.

Related Commands

- `opt_signoff`
- `time_design_signoff`

opt_signoff_max_runtime

```
opt_signoff_max_runtime integer
```

Default: 0

Data_type: int, read/write

Specifies the time out period, in minutes, after which `opt_signoff` will exit the session.

Note: If set, this attribute must be applied before `opt_signoff`.

Related Commands

- `opt_signoff`
- `time_design_signoff`

opt_signoff_max_slack

```
opt_signoff_max_slack value
```

Default: `0`

Data_type: double, read/write

Specifies the maximum slack value to be used during ECO Timing DB generation when using PBA. Setting a positive `opt_signoff_max_slack` value forces the software to retime positive GBA slack paths so that ECO optimization engine can benefit from an extra PBA Setup slack credit when doing Hold fixing or Power recovery. A positive `opt_signoff_max_slack` value implies larger runtime during ECO Timing DB generation, but the software will retime only as many paths as what is specified through the `opt_signoff_max_paths` attribute.

- **Note**:
- This attribute is only active when the `opt_signoff_retime` attribute is set.
- If set, this attribute must be applied before `time_design_signoff`.

Related Commands

- `opt_signoff`
- `time_design_signoff`

opt_signoff_num_report_paths

```
opt_signoff_num_report_paths integer
```

Default: `50`

Data_type: int, read/write

Specifies the number of paths printed after signoff timing analysis per view.

Note: If set, this attribute must be applied before `time_design_signoff`.

Related Commands

- `opt_signoff`
- `time_design_signoff`

opt_signoff_nworst

```
opt_signoff_nworst integer
```

Default: `-1`

Data_type: int, read/write

Specifies the nworst control for path-based analysis.

Note:

- This attribute is only active when the `opt_signoff_retime` attribute is set.
- If set, this attribute must be applied before `time_design_signoff`.

Related Commands

- `opt_signoff`
- `time_design_signoff`

opt_signoff_optimize_core_only

```
opt_signoff_optimize_core_only {true | false}
```

Default: `false`

Data_type: bool, read/write

Performs only reg2reg optimization, when set to `true`.

Note:

- This attribute is not applicable for DRV optimization.
- If set, this attribute must be applied before `opt_design`.

Related Commands

- `opt_signoff`
- `time_design_signoff`

opt_signoff_optimize_replicated_modules

```
opt_signoff_optimize_replicated_modules {true | false}
```

Default: false

Data_type: bool, read/write

Allows the software to perform timing optimization inside replicated hierarchies while generating only one ECO file per sets of clones. The cell names of the replicated hierarchies have to be listed in a file pointed by the `opt_signoff_partition_list_file` attribute.

Note: If set, this attribute must be applied before `opt_signoff`.

Related Commands

- `opt_signoff`
- `time_design_signoff`

opt_signoff_optimize_sequential_cells

```
opt_signoff_optimize_sequential_cells {true | false}
```

Default: `true`

Data_type: bool, read/write

Allows the software to perform timing optimization for sequential cells.

Note: If set, this attribute must be applied before `opt_signoff`.

Related Commands

- `opt_signoff`
- `time_design_signoff`

opt_signoff_partition_list_file

```
opt_signoff_partition_list_file filename
```

Default: "" (empty string)

Data_type: string, read/write

Specifies the file containing list of partitions in the design.

Note: If set, this attribute must be applied before `opt_signoff`.

Related Commands

- `opt_signoff`
- `time_design_signoff`

opt_signoff_pba_eco_route

```
opt_signoff_pba_eco_route {true | false}
```

Default: `false`

Data_type: bool, read/write

When set to `true`, enables `route_eco` to use accurate path-based analysis (PBA) timing during the ECO signoff timing/power closure flow. This attribute must be applied at the start of the `opt_signoff/time_design_signoff flow`.

Related Commands

- opt_signoff
- time_design_signoff

opt_signoff_pba_effort

```
opt_signoff_pba_effort {medium | high}
```

Default: `medium`

Data_type: enum, read/write

When high effort mode is specified, all the optimization engines will perform more aggressive timing closure or area/power recovery with respect to path-based (PBA) timing.

Note: If set, this attribute must be applied before `opt_signoff` and `time_design_signoff`.

Related Commands

- opt_signoff
- time_design_signoff

opt_signoff_post_mask

```
opt_signoff_post_mask {true | false}
```

Default: `false`

Data_type: bool, read/write

Enables the timing closure engine to perform the following netlist changes without touching the base layer mask:

- Inserts the Gate Array buffers or an inverter pair in place of existing Gate Array filler cells.
- Resizes regular cells to Gate Array cells. It also resizes Gate Array cells to Gate Array cells.
- Deletes a regular cell.

The software automatically identifies the Gate Array cells by checking which library instances have the same SITE name as the Gate Array filler cells specified by the user. Here, empty spaces are filled up by the Gate Array filler cells to ensure 100% placement density.

Note:

- This is supported by DRV, Setup, and Hold fixing.
- When `add_tieoffs_cells{{tieLo_name tieHi_name}}` setting is used, the Metal ECO flow will insert the TieLo cells to connect dangling input pins, in case the existing TieLo cells in the floorplan are extremely far or too overloaded.

Related Commands

- `opt_signoff`
- `add_tieoffs` Category Attributes

opt_signoff_post_mask_decap_cells

`opt_signoff_post_mask_decap_cells` *`list_of_decap_cells`*

Default: `""`

Data_type: string, read/write

Lists all the programmable decap cells that are allowed to be removed in post-mask mode.

Note: This attribute is applicable in the post-mask mode only.

Related Commands

- `opt_signoff`
- `time_design_signoff`

opt_signoff_post_mask_filler_cells

`opt_signoff_post_mask_filler_cells` *`list_of_filler_cells`*

Default: `""`

Data_type: string, read/write

Lists all the programmable filler cells that are allowed to be removed/reinserted in post-mask mode.

Note: This attribute is applicable in the post-mask mode only.

Related Commands

- opt_signoff
- time_design_signoff

opt_signoff_post_mask_gate_array_filler_cells

`opt_signoff_post_mask_gate_array_filler_cells` *`list_of_Gate_Array_filler_cells_name`*

Default: `""`

Data_type: string, read/write

Enables the software to list all the Gate Array filler cells that can be deleted or inserted back.

Note: This attribute is applicable in the Metal ECO mode only.

Related Commands

- opt_signoff
- time_design_signoff

opt_signoff_post_mask_logic_cells

`opt_signoff_post_mask_logic_cells` *`list_of_regular_cells`*

Default: `""`

Data_type: string, read/write

Lists all the programmable regular logic/ECO cells that are allowed to be inserted in place of filler/decap cells during optimization in post-mask mode.

Note: This attribute is applicable in the post-mask mode only.

Example:

The following set of commands allow Hold timing closure in post-mask mode using programmable filler cells:

```
set_db opt_signoff_post_mask true
```

```
set_db opt_signoff_post_mask_filler_cells {ln10_ecofiller40x … ln10_ecofiller4x}
set_db opt_signoff_post_mask_decap_cells {ln10_ecofillsgcap40x … ln10_ecofillsgcap4x}
set_db opt_signoff_post_mask_logic_cells {ln10_ecodlybt01 ln10_ecobuft04 …
ln10_ecoand02 … ln10_ecoinv04 …}
opt_signoff -hold
```

Related Commands

- `opt_signoff`
- `time_design_signoff`

opt_signoff_post_sta_tcl

```
opt_signoff_post_sta_tcl filename
```

Default: "" (empty string)

Data_type: string, read/write

Specifies the file containing the TCL code that will be executed immediately after signoff timing analysis per view.

Note:

- These TCL codes should follow Tempus syntax.
- If set, this attribute must be applied before `time_design_signoff`.

Related Commands

- `opt_signoff`
- `time_design_signoff`

opt_signoff_power_aware

```
opt_signoff_power_aware {true / false}
```

Default: false

Data_type: bool, read/write

Enables power awareness in all optimizers and transforms.

Tempus ECO provides the ability to take power as part of the overall cost function when doing timing closure.

When this attribute is enabled before the start of an optimization session, the current optimization and any subsequent optimizations (if running in incremental mode) checks for power changes for all optimizers and transforms during any changes in the design. In any case, the first priority remains getting the best final timing but enabling this option will get there with possibly slighter ECO although resulting in lesser final power.

For example, during to fix a Setup timing path violation, by default the tool would potentially choose to size one instance to a high leakage cell, while with `opt_signoff_power_aware` mode, the tool would try to get to same timing by doing multiple sizing to non-high leakage cells.

Note

- This feature requires power activity to be annotated on the netlist with `report_power` command, either through external activity file (recommended) or using propagated activity based on some default toggling rate.
- This feature honors the `opt_signoff_power_opt_focus` attribute, which means that the user can decide to force the tool in reducing more total or leakage or dynamic power increase.

Related Commands

- `opt_signoff`
- `time_design_signoff`

opt_signoff_power_opt_focus

`opt_signoff_power_opt_focus total | leakage | dynamic | leakageOnly | dynamicOnly`

Default: `total`

Data_type: enum, read/write

Specifies whether a given netlist should be optimized only for leakage or dynamic power reduction, or more for leakage or dynamic power while still reducing the overall total power during total power optimization.

The following options are available:

- `total`: The power optimization engine focuses on getting the optimal total power in the design.

- `leakage`: The power optimization engine optimizes total power but focuses more on leakage power reduction than on dynamic power reduction.
- `dynamic`: The power optimization engine optimizes total power but focuses more on dynamic power reduction than on leakage power reduction.
- `leakageOnly`: The power optimization engine focuses only on leakage power reduction.
- `dynamicOnly`: The power optimization engine focuses only on dynamic power reduction.

Note:

- This parameter is honored by `opt_signoff -power` and also by `opt_signoff -drv/-setup/-hold` when the `opt_signoff_power_aware` attribute is set to `true`.
- If set, this attribute must be applied before `opt_signoff`.

Related Commands

- `opt_signoff`
- `time_design_signoff`

opt_signoff_pre_sta_tcl

```
opt_signoff_pre_sta_tcl filename
```

Default: "" (empty string)

Data_type: string, read/write

Specifies the file containing the TCL code that will be executed immediately before signoff timing analysis per view.

Note: If set, this attribute must be applied before `time_design_signoff`.

Related Commands

- `time_design_signoff`

opt_signoff_prefix

```
opt_signoff_prefix prefix
```

Default: ESO

Data_type: string, read/write

Specifies the prefix to be assigned to the new instances inserted.

Note: If set, this attribute must be applied before `opt_signoff`.

Related Commands

- `opt_signoff`
- `time_design_signoff`

opt_signoff_preserve_filler

```
opt_signoff_preserve_filler  {true | false}
```

Default: `true`

Data_type: bool, read/write

When set to `true`, saves and restores filler states during signoff timing optimization. This helps improve the runtime as it saves the time taken to re-insert all the fillers.

Note: If set, this attribute must be applied before `opt_signoff`.

Related Commands

- `opt_signoff`
- `time_design_signoff`

opt_signoff_read_eco_opt_db

```
opt_signoff_read_eco_opt_db dir_name
```

Default: "" (empty string)

Data_type: string, read/write

Specifies the path of directory where previously generated database files are stored.

Note: If set, this attribute must be applied before `opt_signoff`.

Related Commands

- opt_signoff
- time_design_signoff

opt_signoff_read_irdrop_db

```
opt_signoff_read_irdrop_db dir_name
```

Default: "" (empty string)

Data_type: string, read/write

Specifies the path of directory where previously generated IR drop database files are stored.

Note: If set, this attribute must be applied before `opt_signoff`.

Related Commands

- opt_signoff
- time_design_signoff

opt_signoff_resize_inst

```
opt_signoff_resize_inst {true | false}
```

Default: `true`

Data_type: bool, read/write

Enables or disables resizing transform.

Note: If set, this attribute must be applied before `opt_signoff`.

Related Commands

- opt_signoff
- time_design_signoff

opt_signoff_retime

```
opt_signoff_retime {none |aocv | path_slew_propagation | aocv_path_slew_propagation}
```

Default: `none`

Data_type: enum, read/write

Selects the retime mode for path-based analysis.

Note: If set, this attribute must be applied before `opt_signoff` and `time_design_signoff`.

Related Commands

- `opt_signoff`
- `time_design_signoff`

opt_signoff_retime_mode

```
opt_signoff_retime_mode {none | path | exhaustive}
```

Default: `path`

Data_type: string, read/write

Retimes mode for path based analysis.

Related Commands

- `opt_signoff`
- `time_design_signoff`

opt_signoff_retime_depth

```
opt_signoff_retime_depth {default | infinite}
```

Default: `default`

Data_type: string, read/write

Retimes depth for exhaustive path based analysis.

Related Commands

- opt_signoff
- time_design_signoff

opt_signoff_retime_depth_early

```
opt_signoff_retime_depth_early {default | infinite}
```

Default: `default`

Data_type: string, read/write

Retimes depth for exhaustive path based early analysis.

Related Commands

- opt_signoff
- time_design_signoff

opt_signoff_routing_congestion_aware

```
opt_signoff_routing_congestion_aware {true | false}
```

Default: `false`

Data_type: bool, read/write

Checks routing congestion before adding buffers.

Note: If set, this attribute must be applied before `opt_signoff`.

Related Commands

- opt_signoff
- time_design_signoff

opt_signoff_select_hold_endpoints

```
opt_ signoff_select_hold_endpoints  file_name
```

Default: " " (empty string)

Data_type: string, read/write

Specifies a file containing the list of selected endpoints for hold fixing.

Note: When including endpoints, the `opt_signoff_optimize_core_only` attribute must be set to `false`.

Related Command

- `opt_signoff`

opt_signoff_select_drv_net_file

```
opt_signoff_select_drv_net_file file_name
```

Default: " " (empty string)

Data_type: string, read/write

Specifies a file containing the list of nets to be optimized during DRV fixing. The fixing of max_transition and max_capacitance violations occurs only on the specified nets and the other violating nets are ignored. The file specified includes one net per line.

Note:

- Wild cards and regular expressions are not supported in this file.
- If set, this attribute must be applied before `opt_signoff`.

Related Command

- `opt_signoff`

opt_signoff_select_drv_pin_file

```
opt_signoff_select_drv_pin_file file_name
```

Default: `" "` (empty string)

Data_type: string, read/write

Specifies a file containing the list of pins to be optimized during DRV fixing. The fixing occurs only on the specified instance pins and the other violating pins are ignored. The file specified includes one pin per line.

Note:

- Wild cards and regular expressions are not supported in this file.
- If set, this attribute must be applied before `opt_signoff`.

Related Command

- `opt_signoff`

opt_signoff_select_setup_endpoints

```
opt_ signoff_select_setup_endpoints  file_name
```

Default: `" "` (empty string)

Data_type: string, read/write

Specifies a file containing the list of selected endpoints for setup fixing.

Note:

- When including endpoints, the `opt_signoff_optimize_core_only` attribute must be set to `false`.
- If set, this attribute must be applied before `opt_signoff`.

Related Command

- `opt_signoff`

opt_signoff_setup_recovery

```
opt_signoff_setup_recovery {true | false}
```

Default: `false`

Data_type: bool, read/write

Specifies power-aware setup recovery for setup fixing.

Note: If set, this attribute must be applied before `opt_signoff`.

Related Commands

- `opt_signoff`
- `time_design_signoff`

opt_signoff_setup_target_slack

`opt_signoff_setup_target_slack` *`slack`*

Default: `0.0`

Data_type: double, read/write

Specifies a target slack value in nanoseconds for Setup timing optimization.

Note: If set, this attribute must be applied before `opt_signoff`.

Related Commands

- `opt_signoff`
- `time_design_signoff`

opt_signoff_setup_xtalk_delta_threshold

`opt_signoff_setup_xtalk_delta_threshold` *`value_in_ns`*

Default: `0.3 ns`

Data_type: double, read/write

Fixes the nets with the crosstalk delta delay value equal to or greater than a specified threshold value for setup views.

Note: If set, this attribute must be applied before `opt_signoff`.

Related Commands

- opt_signoff
- time_design_signoff

opt_signoff_setup_xtalk_slack_threshold

opt_signoff_setup_xtalk_slack_threshold *value_in_ns*

Default: 1000 ns

Data_type: double, read/write

Fixes the nets with the slack threshold value equal to or less than a specified threshold value for setup views.

Note: If set, this attribute must be applied before opt_signoff.

Related Commands

- opt_signoff
- time_design_signoff

opt_signoff_skip_drv_net_file

opt_signoff_skip_drv_net_file *file_name*

Default: " " (empty string)

Data_type: string, read/write

Specifies a file containing the list of nets to be excluded during DRV fixing. The fixing of max_transition and max_capacitance violations occurs on nets other than specified nets. The file specified includes one net per line.

Note:

- Wild cards and regular expressions are not supported in this file.
- If set, this attribute must be applied before opt_signoff.

Related Command

- opt_signoff

opt_signoff_set_hold_endpoints_margin

```
opt_signoff_set_hold_endpoints_margin file_name
```

Default: " " (empty string)

Data_type: string, read/write

Specifies endpoint-specific margin for hold fixing.

Note: If set, this attribute must be applied before `opt_signoff`.

Related Command

- opt_signoff

opt_signoff_set_setup_endpoints_margin

```
opt_signoff_set_setup_endpoints_margin file_name
```

Default: " " (empty string)

Data_type: string, read/write

Specifies endpoint-specific margin for setup fixing.

Note: If set, this attribute must be applied before `opt_signoff`.

Related Command

- opt_signoff

opt_signoff_swap_inst

```
opt_signoff_swap_inst {true | false}
```

Default: `true`

Data_type: bool, read/write

Enables/disables VT swap resizing. The setup/leakage optimizers do not honor this attribute and always perform VT-swapping.

Note: If set, this attribute must be applied before `opt_signoff`.

Related Commands

- `opt_signoff`
- `time_design_signoff`

opt_signoff_verbose

```
opt_signoff_verbose {true | false}
```

Default: `false`

Data_type: bool, read/write

Specifies whether the command should output log file reporting with verbosity or not.

Note: If set, this attribute must be applied before `opt_signoff`.

Related Commands

- `opt_signoff`
- `time_design_signoff`

opt_signoff_write_eco_opt_db

```
opt_signoff_write_eco_opt_db dir_name
```

Default: `ecoTimingDB`

Data_type: string, read/write

Specifies a path of directory where generated database files are kept.

Note: If set, this attribute must be applied before `time_design_signoff`.

Related Commands

- opt_signoff
- time_design_signoff

opt_virtual

```
opt_virtual
[-help]
[-out_dir dir_name]
[-timing_reports]
```

This command is based on the gigaOpt infrastructure and it provides prototyping capability at several steps of the hierarchical flow to specify and refine the top-level timing constraints and floorplan. It performs, at preRoute stages, fast virtual optimization using buffering and driver resizing. This command updates the timing graph without modifying the netlist. Using gigaOpt infrastructure, the command has the following capabilities:

- Provides a good QOR (AAE DC, gigaOpt global optimizer, and DRV-based)
- Run in multi-threaded mode
- Supports masters and clones

Parameters

`-help`	Outputs a brief description that includes type and default information for each `opt_virtual` parameter. For a detailed description of the command and all of its parameters, use the man command: `man opt_virtual`.
`-out_dir` *dir_name*	Specifies the name of the directory where the software writes timing reports. This parameter is used only when the `-timing_reports` parameter is specified. *Data_type*: string, optional
`-timing_reports`	Generates timing reports that reflect virtual optimization. *Data_type*: bool, optional

opt Category Attributes

The root attributes in the `opt` category are:

- opt_activity_refresh_args
- opt_add_always_on_feed_through_buffers
- opt_add_insts
- opt_add_ports
- opt_add_repeater_report_failure_reason
- opt_all_end_points
- opt_allow_only_cell_swapping
- opt_allow_multi_bit_on_flop_with_sdc
- opt_area_recovery
- opt_area_recovery_setup_target_slack
- opt_concatenate_default_and_user_prefixes
- opt_consider_routing_congestion
- opt_constant_inputs
- opt_constant_nets
- opt_delete_insts
- opt_detail_drv_failure_reason
- opt_detail_drv_failure_reason_max_num_nets
- opt_down_size_insts
- opt_drv_margin
- opt_enable_data_to_data_checks
- opt_enable_podv2_clock_opt_flow
- opt_enable_restructure
- opt_drv
- opt_drv_with_miller_cap

- opt_fix_fanout_load
- opt_hold_allow_overlap
- opt_hold_allow_resize
- opt_hold_allow_setup_tns_degradation
- opt_hold_ignore_path_groups
- opt_hold_cells
- opt_hold_on_excluded_clock_nets
- opt_hold_slack_threshold
- opt_hier_add_antenna_cell
- opt_hier_opt_stage
- opt_high_effort_cells
- opt_hold_target_slack
- opt_honor_density_screen
- opt_honor_fences
- opt_leakage_to_dynamic_ratio
- opt_max_density
- opt_max_length
- opt_move_insts
- opt_multi_bit_combinational_mode
- opt_multi_bit_combinational_opt
- opt_multi_bit_combinational_merge_timing_effort
- opt_multi_bit_combinational_split_timing_effort
- opt_multi_bit_flop_merge_bank_label_inference
- opt_multi_bit_flop_merge_timing_effort
- opt_multi_bit_flop_opt
- opt_multi_bit_flop_name_prefix
- opt_multi_bit_flop_name_separator

- opt_multi_bit_flop_name_suffix
- opt_multi_bit_flop_reorder_bits
- opt_multi_bit_flop_split_report_failure_reason
- opt_multi_bit_flop_split_timing_effort
- opt_multi_bit_unused_bits
- opt_multi_bit_unused_bit_count
- opt_new_inst_prefix
- opt_new_net_prefix
- opt_one_pass_lec
- opt_podv2_flow_effort
- opt_pin_swapping
- opt_post_route_allow_overlap
- opt_post_route_area_reclaim
- opt_post_route_art_flow
- opt_post_route_check_antenna_rules
- opt_post_route_drv_recovery
- opt_post_route_fix_clock_drv
- opt_post_route_fix_glitch
- opt_post_route_fix_si_transitions
- opt_post_route_hold_recovery
- opt_post_route_report_si_transitions
- opt_post_route_setup_recovery
- opt_power_effort
- opt_pre_route_ndr_aware
- opt_preserve_all_sequential
- opt_preserve_hpin_function
- opt_remove_redundant_insts

- opt_resize_flip_flops
- opt_resize_level_shifter_and_iso_insts
- opt_resize_power_switch_insts
- opt_flop_pins_report
- opt_flops_report
- opt_sequential_genus_restructure_report_failure_reason
- opt_setup_target_slack
- opt_target_based_opt_file
- opt_target_based_opt_hold_file
- opt_target_based_opt_file_only
- opt_tied_inputs
- opt_time_design_compress_reports
- opt_time_design_expanded_view
- opt_time_design_num_paths
- opt_time_design_report_net
- opt_time_design_vertical_timing_summary
- opt_unfix_clock_insts
- opt_skew
- opt_skew_apply_delay_limits_to_full_flow
- opt_skew_ccopt
- opt_skew_post_route
- opt_skew_pre_cts
- opt_skew_delay_pre_cts
- opt_skew_macro_only
- opt_skew_max_allowed_delay
- opt_skew_min_allowed_delay
- opt_skew_no_boundary

- opt_verbose

To set the value of any root attribute, use the `set_db` command:

```
set_db attribute_name value
```

To get the current value of any root attribute, use the `get_db` command:

```
get_db attribute_name
```

To get the current value of all attributes in the `opt` category, use the following command:

```
get_db -category opt
```

opt_activity_refresh_args

```
opt_activity_refresh_args "list_of_arguments"
```

Default: `""`

Data_type: string

Specifies the list of simulation options to be passed for Joules Xreplay. The attribute accepts all arguments of the `xreplay_stimulus` command.

The following arguments can be specified using this attribute:

- `[-xrun_path]` Specify xrun binary path to run Xcelium for replay simulation.
- `[-rtl_stim]` Specify the input RTL stimulus file.
- `[-stim_format]` Suboption of `-rtl_stim`, provides format of rtl stimulus.
- `[-dut_instance]` Suboption of `-rtl_stim`, provides dut hierarchy in rtl stimulus.
- `[-start]` Specify start time to run replay for particular stimulation time duration, for example 24ns. Default is simulation start time.
- `[-end]` Specify end time to run replay for stimulation time duration, for example 100ns. Default is simulation end time.
- `[-verilog_model_files]` Specify list of library verilog models
- `[-stim_annotation]` Honor stim annotation of only the specified objects.
- `[-joules_setting]` Specify the Tcl file to be sourced before xreplay_stim command
- `[-out]` Specify the output replayed gate-level stimulus file. Default is

joulesWorkDir/replayed_stim.shm

- `[-reset_activity_before_readback]` Will reset the previous activity before reading the xreplayed activity file.
- `[-delay_mode none | zero | unit | punit | delay | pdelay] (default: none)`

For setting the arguments: `set_db opt_activity_refresh_args "<args>"`

For querying the arguments: `get_db opt_activity_refresh_args`

opt_add_always_on_feed_through_buffers

```
opt_add_always_on_feed_through_buffers {true | false}
```

Default: false

Data_type: bool

Used by `opt_design` and PD-aware `add_repeater_by_rule`. When this attribute is set to `true`, `opt_design` will use always-on buffers in both pre-Route and post-Route for all the optimization steps; and `add_repeater_by_rule` will use always-on buffer for repeater insertion. It should be used for aggressive always-on buffering during optimization.

The default `opt_design` behavior is to use always-on buffers only during pre-Route DRV fixing.

Related Commands

- `opt_design`
- `add_repeater_by_rule`

opt_add_insts

```
opt_add_insts {true | false}
```

Default: true

Data_type: bool

Specifies whether or not timing optimization can add instances.

Related Command

- opt_design

opt_add_ports

```
opt_add_ports {true | false}
```

Default: `true`

Data_type: bool

Controls whether or not timing optimization adds ports to the lower-level hierarchies of the design. Prohibiting the software from adding ports (when this attribute is set to false) might not be optimal because buffers are not allowed to be added exactly where needed; however, this preserves hierarchical boundaries of the design so that generated test vectors are still valid, and module-by-module verification is possible.

Related Command

- opt_design

opt_add_repeater_report_failure_reason

```
opt_add_repeater_report_failure_reason {true | false}
```

Default: `false`

Data_type: bool

When set to `true`, enables the printing of detailed failure reasons for each net for the `add_repeater_by_rule` command. By default, the report will be generated in the `timingReports` directory. If the directory is not available, it will be created. For example,

```
<run_dir>/timingReports/ARRFailureReason/ARRRpt.preRoute.1
```

Use Model

```
set_db  opt_add_repeater_report_failure_reason true

add_repeater_by_rule -include_nets_file ./net.list -rule $scr/rule -pre_route -
report_ignored_nets ignore_nets
```

Note the following:

- The ignored nets report is appended at the beginning of the failure reason report file. If "`-report_ignored_nets IGNORE_NET_REPORT_NET_FILE_NAME`" is specified, then the ignored nets report is also saved to the file "`IGNORE_NET_REPORT_NET_FILE_NAME`".
- For multiple calls of ARR with different set of nets and constraints, equal number of failure reports will be generated under the `ARRFailureReason` directory. A suffix, "`_<n>`" will be added, with n being the nth call of ARR. For example, three calls of ARR will lead to the generation of the following three files:

  `<run_dir>/timingReports/ARRFailureReason/ ARRRpt.preRoute.1`

  `<run_dir>/timingReports/ARRFailureReason/ ARRRpt.preRoute.2`

  `<run_dir>/timingReports/ARRFailureReason/ ARRRpt.preRoute.3`
- The default mode of ARR reporting is pre_route. If you specify `-post_route` or `-along_route`, the following will be appended in the report names:`<run_dir>/timingReports/ARRFailureReason/ ARRRpt.postRoute.1`

  `<run_dir>/timingReports/ARRFailureReason/ ARRRpt.alongRoute.1`

Note: The behavior of the parameter, `-report_ignored_nets`, remains unchanged. It writes the reports with user-specified file name under the run directory.

Related Command

- `opt_design`

opt_all_end_points

```
opt_all_end_points {true | false}
```

Default: false

Data_type: bool

Runs total negative slack optimization in addition to the worst negative slack optimization and does so on all end points to ensure a high quality timing result.

This feature is automatically activated while running either of the `set_db design_flow_effort standard | extreme` timing-closure flows. So, despite the default being `false` it is overwritten by the `design_flow_effort` attribute setting.

Note: When this attribute is set to `false`, there may still be some fast TNS optimization done during `pre_route` optimization but there is no effect on `post_route` setup optimization. It also has no

effect on hold optimization, which means that regardless of this setting, all hold violating paths will always be optimized.

Related Commands

- `opt_design`
- `place_design`

opt_allow_only_cell_swapping

```
opt_allow_only_cell_swapping {true | false}
```

Default: false

Data_type: bool, read/write

Forces the post-route optimization to perform only same-size Vth swapping for timing improvements. This attribute works best on multi-Vth library designs.

Note: Use this attribute for reoptimizing the timing in specific cases where routing should remain unchanged.

Related Command

- `opt_design`

opt_allow_multi_bit_on_flop_with_sdc

```
opt_allow_multi_bit_on_flop_with_sdc {true | false | merge_only | split_only}
```

Default: `true`

Data_type: bool, read/write

When this attribute is set to `true`, or by default, the software will not honor the SDC constraints while merging or splitting of flops in the multi-bit flip flop flow. Set this attribute to `false` to honor SDC constraints while merging or splitting of flops.

Use `merge_only` to ignore the SDC constraints while merging but honor for splitting. Use `split_only` to ignore the SDC constraints while splitting but honor for merging.

Note: When this attribute is set to `true`, the software will do merge/split even if there is SDC on D/Q

pins and the SDC will be copied to the corresponding D/Q pins in the new flop.

But if there are SDC constraints on SE/SI pin, the merging becomes more complicated. For example, a single flop A and flop B have 1 SE pin and D/Q pin. When merging is done, the SDC on D/Q pin will be copied to the new D0/Q0 and D1/Q1, and the SDC information will not be lost. But in the new 2-bit flop, there will be only one SE pin. This raises the issue of which SDC should be copied into the new flop. In this case, the software will check the SE pin SDC in flop A and flop B. If they are compatible, then it will do merging. Otherwise, the merging will not be done, even if the attribute is set to `true`.

Related Command

- `opt_design`

opt_area_recovery

```
opt_area_recovery {false | true | default}
```

Default: `default`

Data_type: enum, read/write

Controls whether timing optimization creates additional space by downsizing gates or deleting buffers, while maintaining worst slack and total negative slack.

This attribute can be used in three modes: `default`, `true`, and `false`.

- `default`: By default, area reduction is always called after global optimation and after the second TNS optimization. During the rest of `opt_design`, area reduction is called depending on the density level of the current design.
- `true`: When this attribute is set to `true`, area reduction is always called before the global optimization stage, after the global optimization stage, and after the second TNS optimization. During the rest of `opt_design`, area reduction is called depending on the density level of the current design.
- `false`: No area reduction is performed during `opt_design`.

Note: This attribute is used to reclaim area during pre-Route optimization. For reclaiming area during post-Route optimization, use the `opt_post_route_area_reclaim` attribute.

Related Command

- opt_design

opt_area_recovery_setup_target_slack

```
opt_area_recovery_setup_target_slack slack
```

Default: 0.0s

Data_type: double, read/write

Specifies a user-defined target slack for area reclaim vs. timing. When a slack value is specified, the software prevents area reclaim optimization from creating or working on near critical paths. Typically, you can set this attribute to 0.1 with opt_setup_target_slack attribute at the default value of 0.0. This implies that the area reclaim will not work on any paths less than 0.1ns.

Note that the value for opt_setup_target_slack attribute takes higher priority. For example, if opt_area_recovery_setup_target_slack is set to -0.1 and opt_setup_target_slack is at 0.0 then setup timing will not be allowed to degrade past 0ps and a warning will be issued.

Similarly, if say opt_area_recovery_setup_target_slack is at +0.1 and opt_setup_target_slack is set to +0.2 then timing will target +0.2 and a warning will be issued.

You need to set this attribute before any area reclaim is done.

Related Command

- opt_design

opt_concatenate_default_and_user_prefixes

```
opt_concatenate_default_and_user_prefixes {true | false}
```

Default: true

Data_type: bool, read/write

Controls whether to combine the default prefix for cells and nets added by timing optimization with the prefixes specified by opt_new_net_prefix and opt_new_inst_prefix attributes. By default, the software combines the specified prefixes with the default prefix. Specify false for this attribute to ignore the default prefix and use only the prefix specified by opt_new_net_prefix and opt_new_inst_prefix attributes.

For example, if the default prefix is PSN_000234 and you set the following attributes:

```
opt_concatenate_default_and_user_prefixes true

opt_new_net_prefix POSTROUTE_N

opt_new_inst_prefix POSTROUTE_C
```

The software uses the following prefixes:

- For cells added by optimization: `POSTROUTE_C_PSN_00234`
- For nets added by optimization: `POSTROUTE_N_PSN_00234`

If the default prefix is `PSN_000234` and you set the following attribute:

```
opt_concatenate_default_and_user_prefixes false

opt_new_net_prefix POSTROUTE_N

opt_new_inst_prefix POSTROUTE_C
```

The software uses the following prefixes:

- For cells added by optimization: `POSTROUTE_C`
- For nets added by optimization: `POSTROUTE_N`

Note: If you specify `false` for this attribute, you must also specify `opt_new_inst_prefix` and `opt_new_net_prefix` attributes. In addition, you must specify a different prefix for instances and nets.

Related Command

- `opt_design`

opt_consider_routing_congestion

```
opt_consider_routing_congestion {auto | false | true}
```

Default: `auto`

Data_type: enum, read/write

Checks the routing congestion at the post-Route stage during optimization of setup, hold, drv, or glitch. This attribute also impacts post-CTS hold fixing. By default, this attribute is set to `auto` for both `opt_design -post_route` and `opt_design -post_route -hold` options. This attribute helps improve NR DRC's as well as Timing jumps caused by routing detours on congested designs.

Related Command

- opt_design

opt_constant_inputs

```
opt_constant_inputs {true | false}
```

Default: `false`

Data_type: bool, read/write

Forces optimization of instances with constant inputs, even if it worsens the objective gain - in terms of area, power, and delay.

Related Command

- opt_design

opt_constant_nets

```
opt_constant_nets {true | false}
```

Default: true

Data_type: bool, read/write

Specifies whether or not timing optimization optimizes constant nets.

Related Command

- opt_design

opt_delete_insts

```
opt_delete_insts {true | false}
```

Default: true

Data_type: bool, read/write

Specifies whether or not timing optimization can delete instances.

Related Command

- opt_design

opt_detail_drv_failure_reason

```
opt_detail_drv_failure_reason {true | false}
```

Default: false

Data_type: bool, read/write

When set to true, enables the printing of detailed DRV failure reason report file that contains the faliure summary with the net name. The log file contains a simple version of failure summary without the net name. In addition, the top number(default 50) of violating nets are printed in separate files along with the violation reasons.

By default, this attribute is set to false, which means that only a simple version of failure summary without the net name is printed in the log file and no other files are created.

Note: If there is no DRV failure during the optimization flow, the failure reason report file will not be generated even when this attribute is set to true.

Related Command

- opt_design

opt_detail_drv_failure_reason_max_num_nets

```
opt_detail_drv_failure_reason_max_num_nets value
```

Default: 50

Data_type: int, read/write

Specifies the number of violating nets for which the DRV failure reasons are to be printed in the report files. The default value is 50, which means failure reasons for 50 violating nets are printed by default. The minimum value that can be specified is 1. The maximum value that can be specified is 9999. If this value is specified, failure reasons for all the violating nets are printed.

For example, if you set this attribute to 100, then the top 100 max tran violating nets, if present, will be printed in file, "xx.maxTran" and the top 100 max cap violating nets, if present, will be printed in file, "xx.maxCap". Also, a file, "xx.maxFanout" may also be generated.

For each file that is generated, the first printed net has the worst violation value. The file is sorted based on violation values while printing.

Related Command

- opt_design

opt_down_size_insts

```
opt_down_size_insts {true | false}
```

Default: true

Data_type: bool, read/write

Specifies whether or not timing optimization can downsize instances.

Related Command

- opt_design

opt_drv_margin

```
opt_drv_margin margin
```

Default: 0.0

Data_type: double, read/write

Scales the maxCap and maxTran constraints according to the *margin* decimal value specified. For example, specifying a value of 0.2 for this attribute tightens the margin (makes it more pessimistic) by 20 percent during optimization and specifying a value of -0.2 relaxes the margin (makes it more optimistic) by 20 percent. Introducing a pessimistic margin improves correlation between the number of DRC violations before and after routing.

Value range: -0.7 ~ 0.7

Related Command

- opt_design

opt_enable_data_to_data_checks

```
opt_enable_data_to_data_checks {true | false}
```

Default: false

Data_type: enum, read/write

Enables data to data checks during optimization. During optimization the data to data timing checks are set to off by default to ensure that the engine focuses on other paths. To allow checks to be visible during optimization you can set this option to true.

Related Command

- `opt_design`

opt_enable_podv2_clock_opt_flow

`opt_enable_podv2_clock_opt_flow {true | false}`

Default: `false`

Data_type: bool, read/write

When set to `true`, enables the place_opt_design V2 (PODV2) and `clock_opt_design` flow.

opt_enable_restructure

```
opt_enable_restructure {true | false}
```

Default: true

Data_type: bool, read/write

Specifies whether or not timing optimization swaps pins and restructures the netlist.

Note: This attribute is not enabled when the `design_flow_effort` attribute is set to `express`.

Related Command

- `opt_design`

opt_drv

```
opt_drv {true | false}
```

Default: `true`

Data_type: bool, read/write

Repairs all nets that are constrained for maximum capacitance violations and maximum transition violations. The capacitance and transition constraint values are specified in the timing constraint file or in the library. If you specify false for this attribute, the software repairs only the paths with timing violations.

Related Command

- `opt_design`

opt_drv_with_miller_cap

```
opt_drv_with_miller_cap {true | false}
```

Default: `false`

Data_type: bool, read/write

When set to `true`, enables ECSM/CCS miller capacitance DRV reporting and fixing for `time_design` and `opt_design`. However, the `timing_report_use_receiver_model_capacitance` attribute must also be set to `true`.

By default, this parameter is set to `false`, which means that miller cap DRV reporting and fixing is disabled.

Related Command

- `opt_design`
- `time_design`

opt_fix_fanout_load

```
opt_fix_fanout_load {true | false}
```

Default: `false`

Data_type: bool, read/write

Forces timing optimization to correct fanout load violations, which can either be maximum load or maximum fanout count violations, depending on the libraries.

Related Command

- `opt_design`

opt_hold_allow_overlap

```
opt_hold_allow_overlap {auto | true | false}
```

Default: `auto`

Data_type: enum, read/write

Allows hold-fixing algorithm to insert buffers with overlap when large enough empty space locations are not available. The placement will then be legalized before ECO routing. Setting this attribute allows the tool to fix hold violations, but might have some negative impact on setup timing.

- `auto`: Hold fixing will run in the "allow overlapping" mode in post-CTS and not in post-Route stage.
- `true`: Hold fixing will run in the "allow overlapping" mode in both post-CTS and post-Route stage.
- `false`: Hold fixing will run in "allow overlapping" mode in both post-CTS and post-Route stage.

Related Command

- `opt_design`

opt_hold_allow_resize

```
opt_hold_allow_resize {true | false}
```

Default: `true`

Data_type: bool, read/write

Controls the resizing capability inside hold optimization. When set to `true` or `false`, resizing is turned "on" or "off" in both post-CTS and post-Route hold optimization.

Related Command

- `opt_design`

opt_hold_allow_setup_tns_degradation

```
opt_hold_allow_setup_tns_degradation {true | false}
```

Default: true

Data_type: bool, read/write

Controls whether the software allows total negative slack to degrade during hold fixing operations or not. By default, the software allows the slack to degrade in order to fix more hold violations. To override the default behavior and ensure that the setup total negative slack is maintained during hold fixing, set this attribute to `false`.

Related Command

- `opt_design`

opt_hold_ignore_path_groups

```
opt_hold_ignore_path_groups {groupAgroupB....}
```

Default: `""` (empty string)

Data_type: string, read/write

Specifies the clock domains or path groups that should be excluded from hold timing optimization. This attribute impacts only `opt_design -post_cts/-post_route -hold`.

Note: To reset this attribute, apply an empty list.

Note: If a violated path is in two path groups and only one of those paths is specified using this attribute, this path is still fixed during hold fixing.

Related Command

- opt_design

opt_hold_cells

```
opt_hold_cells list_of_buffers
```

Default: "" (empty string). This will trigger the default flow.

Data_type: string, read/write

Allows you to force the usage of only a specific list of buffer and delay cells. The *list_of_buffers* contains regular buffers and delay buffer cells. When this list contains at least one cell, use only the specified cells.

Note: This attribute does not support wildcards. The *list_of_buffers* must contain only exact cell names.

Note: When this list of buffers is specified, the opt_design command disables automatic cell filtering, which could impact the runtime/QOR/leakage of opt_design.

Note: The opt_design command uses the specified cells without honoring the dont_use attribute.

Related Command

- opt_design

opt_hold_on_excluded_clock_nets

```
opt_hold_on_excluded_clock_nets {true | false}
```

Default: false

Data_type: bool, read/write

Controls whether or not hold fixing will be performed on excluded clock nets. This attribute is applicable for both post-CTS and post-Route Hold optimization.

Related Command

- opt_design

opt_hold_slack_threshold

```
opt_hold_slack_threshold slack
```

Default: `-1000ns`

Data_type: double, read/write

Allows you to provide a hold slack threshold beyond which the hold timing violations will not be fixed. This is used to avoid fixing out of bound violations at an early stage when the timing constraints and design environment might not be mature enough. The slack value is specified in nano seconds (ns).

Note: When the hold slack threshold value is specified using this attribute, the hold fixing timing table excludes the violations with slack worse than the specified value.

Note: The threshold value specified is for hold WNS.

Related Command

- `opt_design`

opt_hier_add_antenna_cell

```
opt_hier_add_antenna_cell {true | false}
```

Default: false

Data_type: bool, read/write

Adds antenna cells to the interface nets crossing partition boundaries. The antenna violations could not be created due to the partition boundary nets. These violations are seen only when the full design is assembled. Therefore, when this attribute is set to `true`, it will add antenna cells to fix these violations.

opt_hier_opt_stage

```
opt_hier_opt_stage {type {enum pre_cts post_cts unset}}
```

Default: unset

Data_type: enum, read/write

Ensures proper settings are turned on for

corresponding `opt_design` stages. `opt_hier_opt_stage` should be set before `opt_design`, and in cases where module view types are defined, `opt_hier_opt_stage` should be defined after module view types for all partitions (including FlexILMs) are defined.

For example, `opt_hier_opt_stage pre_cts` ensures that `opt_design -pre_cts` honors fence, keep partition boundary port, etc.

Related Command

- `opt_design`

opt_high_effort_cells

```
opt_high_effort_cells list_of_cells
```

Default: `""` (empty string)

Data_type: string, read/write

Specifies the list of cells to be used for high-effort optimization. This attribute is applicable in both pre-Route and post-Route setup optimization.

Note: If the `dont_touch` constraint is set to `true` for the specified cells, then these cells will not be used for high-effort optimization.

Related Command

- `opt_design`

opt_hold_target_slack

```
opt_hold_target_slack holdTargetSlack
```

Default: `0.0`

Data_type: double, read/write

Specifies a target slack value in nanoseconds to use for hold analysis only. During setup violation repair, the setup target slack is defined by the attribute, `opt_setup_target_slack` and the hold target slack value is 0. Generally, you use both the `opt_setup_target_slack` and the `opt_hold_target_slack` attributes for hold analysis. If you specify only `opt_hold_target_slack` attribute, the setup target slack value is 0.

Related Command

- opt_design

opt_honor_density_screen

```
opt_honor_density_screen {true | false}
```

Default: false

Data_type: bool, read/write

Controls whether or not density screens (partial placement blockages used to control congestion) are honored while optimizing timing. By default, this attribute is set to false , which means that the density screens are not honored.

Related Command

- opt_design

opt_honor_fences

```
opt_honor_fences {true | false}
```

Default: false

Data_type: bool, read/write

Specifies that the timing optimization takes fences or region constraints into account. This attribute is honored by each opt_design step. If you optimize the design with this attribute set to true, the placement is legal.

This feature is useful for hierarchical designs. Because this attribute adds constraints to timing optimization, enable it only when needed.

Note: opt_design will not honor very small fences when this attribute is set to true if it would cause fence utilization to be greater than 100 percent.

Related Command

- opt_design

opt_leakage_to_dynamic_ratio

```
opt_leakage_to_dynamic_ratio ratio
```

Default: `1.0`

Data_type: double, read/write

Controls the priority of the power-driven optimization. You can set any value between 0 and 1. Use this attribute along with the `opt_power_effort` attribute.

Some examples of what the value of this attribute indicates are provided below:

- `opt_leakage_to_dynamic_ratio 1.0`
  This is the default value and implies that timing optimization will be purely leakage-power driven optimization.
- `opt_leakage_to_dynamic_ratio 0`
  This implies that timing optimization will be purely dynamic power-driven optimization.
- `opt_leakage_to_dynamic_ratio 0.5`
  This implies that timing optimization will be both dynamic and leakage power-driven and trade-offs may need to be done to get a balance between the leakage and dynamic power optimization.
- `opt_leakage_to_dynamic_ratio 0.1`
  This implies that timing optimization will be both dynamic and leakage power-driven but dynamic power optimization will be favored.
- `opt_leakage_to_dynamic_ratio 0.9`
  This implies that timing optimization will be both dynamic and leakage power-driven but leakage power optimization will be favored.

Related Command

- `opt_design`

opt_max_density

```
opt_max_density value
```

Default: 0.95

Data_type: double, read/write

Specifies the maximum value for density (area utilization). The `opt_design` command does not grow the netlist above this value. When reaching a density close to the value specified by this attribute, `opt_design` has limited optimization capability.

Related Command

- `opt_design`

opt_max_length

```
opt_max_length length
```

Default: `-1.0`

min=-1, max=2147483647

Data_type: double, read/write

Specifies that `opt_design` honors the maxLength rule as any other DRV rule (`max_cap/max_tran/max_fanout`). The `opt_design` command will fix all the nets that are violating this rule and will ensure to not create any long nets that violate it. The value is specified in micrometer.

Note: This attribute implies that more buffers gets inserted in the netlist. This attribute is disabled by default.

Note: During the pre-Route stage, some maxLength violations might still exist after running `opt_design` due to routing topology changes taking place during the flow.

Related Command

- `opt_design`

opt_move_insts

```
opt_move_insts {true | false}
```

Default: `true`

Data_type: bool, read/write

Allows cells in the critical path to move when the software repairs setup violations or optimizes

timing along the critical path, if allowing them to move improves timing.

Related Command

- `opt_design`

opt_multi_bit_combinational_mode

`opt_multi_bit_combinational_mode {auto | power | area}`

Default: `auto`

Data_type: enum, read/write

Enables the user-specific multi-bit combinational cell (MBCC) merging and splitting based on area and power.

`auto`: Enables merging or splitting of MBCCs based on area (*if power effort is none*), otherwise based on power.

`power`: Enables merging or splitting of MBCCs based on power.

`area`: Enables merging or splitting of MBCCs based on area.

Default: `auto`

Related Command

- `opt_design`

opt_multi_bit_combinational_opt

`opt_multi_bit_combinational_opt {false | true | mergeOnly | splitOnly}`

Default: `splitOnly`

Data_type: enum, read/write

Enables and disables the multi-bit combinational flow. This flow is called as a part of the pre-CTS optimization stage. The following settings are available:

- true: enables both merging and splitting of combinational cells
- `false`*:* disables the flow

- `mergeOnly`: only merges the combinational cells
- `splitOnly`: only splits the combinational cells

Related Command

- `opt_design`

opt_multi_bit_combinational_merge_timing_effort

```
opt_multi_bit_combinational_merge_timing_effort {none | low | medium | high}
```

Default: `medium`

Data_type: enum, read/write

Specifies the timing effort-level for multi-bit combinational merging.

- If the attribute is set to `high`, the software merges the combinational cell to maintain or improve the timing.
- If the attribute is set to `low`, the software merges the combinational cell with allowing timing degradation.
- If the attribute is set to `medium`, the software allows merging without/minimal degradation in timing.

Related Command

- `opt_design`

opt_multi_bit_combinational_split_timing_effort

```
opt_multi_bit_combinational_split_timing_effort {none | low | medium | high}
```

Default: `low`

Data_type: enum, read/write

Specifies the timing effort-level for multi-bit combinational splitting.

- If the attribute is set to `low`, the software allows splitting to critical paths only.

- If the attribute is set to `medium`, the splitting happen to critical paths to improve timing.
- If the attribute is set to `high`, the software performs aggressive splitting to improve timing.

Related Command

- `opt_design`

opt_multi_bit_flop_merge_bank_label_inference

`opt_multi_bit_flop_merge_bank_label_inference {true | false}`

Default: `false`

Data_type: bool, read/write

Enables soft merging of flops with the same bank label.

This is a root attribute.

- To set the value of any root attribute, use the `set_db` command:
  `set_db *attribute_name value*`
- To get the current value of any root attribute, use the `get_db` command:
  `get_db *attribute_name*`

Related Commands

- `opt_design`

opt_multi_bit_flop_merge_timing_effort

`opt_multi_bit_flop_merge_timing_effort {low | medium | high}`

Default: `medium`

Data_type: enum, read/write

Specifies the timing effort level for MBFF merging.

- `low` effort setting allows degrading in timing to achieve merging.
- `medium` effort means that merging happens either without or with minimal degradation of timing

after merging.

- `high` effort setting does not allow any degradation and does merging only if it either maintain or improves timing.

Related Command

- `opt_design`

opt_multi_bit_flop_opt

```
opt_multi_bit_flop_opt {true | false | mergeOnly | splitOnly}
```

Default: `splitOnly`

Data_type: bool, read/write

Enables and disables the multi-bit flip flop flow. This flow is called as a part of the pre_CTS optimization stage. Set this attribute to `true` to enable both merging and splitting of flops. Set it to `false` to disable the flow. Set it to `mergeOnly` to only merge flops. By default, the attribute is set to `splitOnly`, which means the flow is enabled only for splitting MBFFs.

Note:

- This option is independent of `opt_enable_restructure, opt_resize_flip_flops`, and `opt_preserve_all_sequential` attributes.
- To ensure sufficient multi-bit conversion ratio, allow more usable multi-bit flop cells from the libraries. Use the command, `report_multi_bit_ffs -all -equivalence` to write out the corresponding multi-bit flop cells to the single-bit flop cells.
- To reduce long runtime in flop merge steps, ensure that the scan enable signal nets connecting to the flops are buffered, which means without max-fanout violations.

Related Command

- `opt_design`

opt_multi_bit_flop_name_prefix

```
opt_multi_bit_flop_name_prefix prefix_name
```

Default: `CDN_MBIT_`

Data_type: string, read/write

Specifies the MBFF name prefix. When single-bit flip-flops are merged into a multi-bit flip-flop, you can use this attribute to specify the prefix for the new merged flip-flop. If no prefix is specified, the name of the merged flip-flop starts with the default name, `CDN_MBIT_`.

Related Command

- `opt_design`

opt_multi_bit_flop_name_separator

`opt_multi_bit_flop_name_separator` *separator_name*

Default: `_MB_`

Data_type: string, read/write

Specifies the flip-flop merge and split name separator. The separator is used while splitting and merging flip-flops. The default separator name is `_MB_` . The use model is detailed below.

Scenario1:

When two or more SBFFs are merged, the separators are used between names of single bit flip-flops. The MBFF is named as; `prefix_bit1_separator_bit2_separator_bit3..._separator_bit#`.

Scenario2:

When an MBFF is split into smaller bits, then each smaller bit is named as; `prefix_bit1_separator_bit2`.

Scenario3:

When an MBFF is split into the original number of single-bit flip-flops, then each SBFF reverts to its original name, which is without prefix and separator such as; `bit1, bit2`...

Consider the example below to understand the above scenarios:

There are four SBFFs: `bit1, bit2, bit3,` and `bit4`.

- With default settings, when they are merged, the following MBFF is generated:
  CDN_MBIT_bit1_**MB**_bit2_**MB**_bit3_**MB**_bit4
  This MBFF has the default prefix and the default separator.
- When the above MBFF is split into two-bit MBFFs, the following is generated:

CDN_MBIT_bit1_**MB**_bit2, and
CND_MBIT_bit3_**MB**_bit4

- When the MBFF is split into the original SBFFs, the names of the SBFFs revert to the original names:
  bit1, bit2, bit3, and bit4

Related Command

- opt_design

opt_multi_bit_flop_name_suffix

```
opt_multi_bit_flop_name_suffix _user_suffix
```

Default: ""

Data_type: string, read/write

Specifies the MBFF name suffix. When this parameter is set, the software adds the user-specified suffix at the end of the multi-bit cell instance name along with the bit-width of the multi-bit cell.

For example, the below command is used to specify _FTREG as suffix:

```
set_db opt_multi_bit_flop_name_suffix  _FTREG

CDN_MBIT_bit1_MB_bit2_FTREG2                    for a 2-bit multibit cell


CDN_MBIT_bit1_MB_bit2_MB_bit3_MB_bit4_FTREG4    for a 4-bit multibit cell
```

Related Command

- opt_design

opt_multi_bit_flop_reorder_bits

```
opt_multi_bit_flop_reorder_bits {false | true | timing | power}
```

Default: false

Data_type: enum, read/write

Enables swap MBFF bits to take advantage of mixed-drive strength (MDS) multi-bit cells. The following settings are possible:

- `false`: the software does not identify the MDS cells and no bit-reordering is done
- `timing`: software identifies and uses the MDS cells for timing optimization
- `power`: software identifies and uses the MDS cells for power optimization
- `true`: software identifies and uses the MDS cells for both timing and power optimization

Note: When this attribute is set to `true`, `power`, or `timing`, a "Bit-strength (strong to weak)" column will be added in the report generated using the `report_multi_bit_ffs -info`, `-all`, and `-cell` parameters.

Related Command

- `opt_design`

opt_multi_bit_flop_split_report_failure_reason

```
opt_multi_bit_flop_split_report_failure_reason {true | false}
```

Default: `true`

Data_type: bool, read/write

When set to `false`, disables the printing of MBFF split failure reasons. By default this parameter is set to `true`, which means the reasons for MBFF instances not splitting are printed. This is the default behavior.

Related Command

- `opt_design`

opt_multi_bit_flop_split_timing_effort

```
opt_multi_bit_flop_split_timing_effort {low | medium | high}
```

Default: `low`

Data_type: enum, read/write

Specifies the timing effort level for MBFF splitting.

- `low` effort setting means splitting will be done only when absolutely necessary for critical paths.
- `medium` effort setting means splitting happens for more critical paths to improve timing and remove critical timing bottlenecks .
- `high` effort setting does aggressive splitting and try to improve timing.

Related Command

- `opt_design`

opt_multi_bit_unused_bits

```
opt_multi_bit_unused_bits {true | false | pre_place | pre_cts}
```

Default: `pre_place`

Data_type: enum, read/write

Specifies whether a mapped multi-bit instance can have one or more unused bits. By default, at `pre_place` stage the software tries to merge the single-bit registers to multi-bit instances without unused bits. If during merging, the software is unable to find a suitable multi-bit lib_cell with a width that matches the left-over bits of a bus, it will choose the closest matching width, which leaves the minimum unused bits and, therefore, increases the multi-bit merging coverage.

When set to `false`, any merging that results in unused (undriven/unloaded) bits is prevented.

When set to `true`, the software tries to merge the single-bit register to multi-bit instances with unused bits at both, `pre_place` and `pre_cts` stages.

When set to `pre_cts`, the software merges the single-bit register to multi-bit instances with unused bits at the pre_cts stage.

Related Command

- `opt_design`

opt_multi_bit_unused_bit_count

```
opt_multi_bit_unused_bit_count integer
```

Default: `0`

Min: `1`

Max: `32`

Data_type: int, read/write

Specifies the maximum number of unused bits to be allowed in each multi-bit cell.

Note: To use this attribute, set the `opt_multi_bit_unused_bits` attribute to `true`.

Related Command

- `opt_design`

opt_new_inst_prefix

```
opt_new_inst_prefix prefix
```

Default: "" (empty string)

Data_type: string, read/write

Specifies a prefix string to add to instances created by timing optimization. For example, if a default prefix is `OFC_000234`, when you use this attribute to specify the prefix, `POSTROUTE_`, the combined prefix added to the instance name is `POSTROUTE_OFC_00234`.

Note: To ignore the default prefix, set the `opt_concatenate_default_and_user_prefixes` attribute to `false`.

Related Command

- `opt_design`

opt_new_net_prefix

```
opt_new_net_prefix  prefix
```

Default: "" (empty string)

Data_type: string, read/write

Specifies a prefix string to add when timing optimization creates nets. For example, if a default prefix is `PSN_000234`, when you use this attribute to specify the prefix, `POSTROUTE_`, the combined prefix added to the net name is `POSTROUTE_PSN_00234`.

Note: To ignore the default prefix, set the `opt_concatenate_default_and_user_prefixes` attribute to false.

Related Command

- `opt_design`

opt_one_pass_lec

```
opt_one_pass_lec {true | false}
```

Default: `false`

Data_type: bool, read/write

When set to `true`, enables the one pass Logical Equivalence Check (LEC) flow. In one pass LEC, you can do a Register Transfer Level (RTL) to netlist comparison between RTL and post-route optimization (PRO) netlist - any place and route (PnR) stage netlist - in a single step.

Related Command

- `opt_design`

opt_podv2_flow_effort

```
opt_podv2_flow_effort {auto | standard | extreme}
```

Default: `auto`

Data_type: enum, read/write

Specifies the effort level for the place_opt_design V2 (PODV2) flow. See `clock_opt_design`.

The following settings are available:

- `auto`: The software follows the `design_flow_effort` attribute setting to perform optimization.
- `standard`: The software performs a fast prects optimization and runtime is less. This option is recommended in the early design stage or in simple designs for PPA estimation.
- `extreme`: The software performs a deep optimization to chase the best PPA. Runtime is high.

opt_pin_swapping

```
opt_pin_swapping {true | false}
```

Default: `true`

Data_type: bool, read/write

Controls whether timing optimization allows pin swapping.

Note: This attribute is not enabled when the `design_flow_effort` attribute is set to `express`.

Related Command

- `opt_design`

opt_post_route_allow_overlap

```
opt_post_route_allow_overlap {true | false}
```

Default: `true`

Data_type: bool, read/write

Controls whether timing optimization allows overlap during `opt_design -post_route`.

Related Command

- `opt_design`

opt_post_route_area_reclaim

```
opt_post_route_area_reclaim {none | setup_aware | hold_and_setup_aware}
```

Default: `none`

Data_type: enum, read/write

This attribute can be used after routing to reclaim area during `opt_design -post_route`. It safely reclaims area while not impacting the timing and DRV violations. You can use the following options:

- `none`: No area reclaim is done. This is the default setting.
- `setup_aware`: Reclaim is done but is not hold aware. This may degrade hold timing significantly but will maintain the setup timing while giving the best possible reclaim results.
- `hold_and_setup_aware`: Reclaim is done but is both hold and setup aware. It will not give as good reclaim results as `setup_aware` alone. But it will mean both setup and hold timing will be maintained.

Related Command

- `opt_design`

opt_post_route_art_flow

```
opt_post_route_art_flow {true | false}
```

Default: false

Data_type: bool, read/write

Ensures that timing optimization applies to ART-based flow. In the ART-based flow, it partially activates the design at the top-level and at the interface path of partition blocks so that there is a reduction in memory usage and run time if the design is large. This attribute marks the core logic inside the partition as dont_touch, so the logic is not optimized.

Note: This attribute requires partition definition information and applies to the post-Route mode only.

Related Command

- `opt_design`

opt_post_route_check_antenna_rules

```
opt_post_route_check_antenna_rules {true | false}
```

Default: `true`

Data_type: bool, read/write

Allows optimization to check/ignore antenna rules for LEF-safe swaps. The default behavior is to check the antenna rules, in order to avoid antenna violations caused by the cell swappings. The LEF files have certain antenna rules defined in them. By default, when doing LEF-safe swaps for leakage power optimization and for LEF-safe timing recovery, it is ensured that the cells have the same antenna rules. If cells do not have the same antenna rules, they are not swapped.

However, if this attribute is set to `false`, in post-Route mode, antenna rules will not be considered when doing LEF-safe swaps. Therefore, setting this attribute to `false` improves timing and leakage but it can potentially lead to routing having antenna violations, which will require a call to the `route_eco` command for cleanup.

Related Command

- `opt_design`

opt_post_route_drv_recovery

```
opt_post_route_drv_recovery {auto | false | true}
```

Default: `auto`

Data_type: enum, read/write

Controls the DRV/glitch recovery step performed when `opt_design -post_route -setup -hold` and `opt_design -post_route -hold` commands have been specified. You can specify the following options:

- `false`: Disables DRV/glitch recovery.
- `auto`: Allows the software to determine when to trigger DRV/glitch recovery. Here recovery will be triggered if any of the DRV's or glitches degrade beyond a certain margin. Note that DRV/glitch recovery is disabled for `opt_design -post_route -hold` by default.
- `true`: Always triggers DRV/glitch recovery if there are any left in the design.

Related Command

- opt_design

opt_post_route_fix_clock_drv

```
opt_post_route_fix_clock_drv {true | false}
```

Default: `true`

Data_type: bool, read/write

Controls whether `clock_post_route_repair` is called or not at the start of post-Route optimization.

By default this attribute is set `true`, which means that `ccopt_post_route` is called to repair clock tree design rule violations (DRVs) that have occurred in the clock tree after routing. These DRVs are most commonly the user-set clock transTime violations. To disable the fixing of DRVs, set the attribute to `false`.

Related Command

- opt_design

opt_post_route_fix_glitch

```
opt_post_route_fix_glitch {true | false}
```

Default: `true`

Data_type: bool, read/write

Determines whether to fix glitch violations during SI optimization or not. To disable fixing of glitch violations, set this attribute to `false`.

Related Command

- opt_design

opt_post_route_fix_si_transitions

```
opt_post_route_fix_si_transitions {true | false}
```

Default: false

Data_type: bool, read/write

Enables fixing of SI-induced slew violations in post-Route optimization.

Note: The transition values will be used for maximum transition violation fixing only and will not affect the timing calculation in the timing engine. This is because the transition effect is considered by the internal SI engine during delay pushout calculation.

Related Command

- `opt_design`

opt_post_route_hold_recovery

```
opt_post_route_hold_recovery  {auto | false | true}
```

Default: `false`

Data_type: enum, read/write

Controls the Hold Recovery step performed when `opt_design -post_route -setup -hold` and `opt_design -post_route -hold` commands have been specified. You can specify the following options:

- `false`: Disables hold recovery.
- `auto`: Allows the software to determine when to trigger setup recovery. Here recovery will be triggered if WNS or TNS degrade beyond a certain margin.
- `true`: Always triggers hold recovery if timing is not met, irrespective of the gain/degradation in post-route timing.

Related Command

- `opt_design`

opt_post_route_report_si_transitions

```
opt_post_route_report_si_transitions {true | false}
```

Default : `false`

Data_type: bool, read/write

Enables reporting of SI slews for DRV maximum transition reporting only. The delay calculation will continue to use base slews for timing analysis commands, such a as `reportTranViolation` , `report_constraint` , `report_delay_calculation` , and `report_timing` . The DRV violations will be reported by `time_design` under the `./timingReports/<design name>.tran file` only and the number of maximum transition violations due to SI slews will be listed in the `time_design` summary report.

Note : The `opt_post_route_report_si_transitions true` setting does not enable fixing of SI slews, this command only enables reporting of them for `time_design` DRV reporting. To enable SI slew fixing, use `opt_post_route_fix_si_transitions true` .

opt_post_route_setup_recovery

```
opt_post_route_setup_recovery  {auto | false | true}
```

Default: `auto`

Data_type: enum, read/write

Controls the Setup Recovery step performed when `opt_design -post_route -setup -hold` and `opt_design -post_route -hold` commands have been specified. You can specify the following options:

- `false`: Disables setup recovery.
- `auto`: Allows the software to determine when to trigger setup recovery. Here recovery will be triggered if WNS or TNS degrade beyond a certain margin.
- `true`: Always triggers setup recovery if timing is not met, irrespective of the gain/degradation in post-route timing.

Related Command

- `opt_design`

opt_power_effort

```
opt_power_effort {none | low | high}
```

Default: `none`

Data_type: enum, read/write

Specifies the power-driven optimization functionality. This attribute provides the effort level for optimizing leakage power and dynamic power in the design. Use this attribute along with the `opt_leakage_to_dynamic_ratio` attribute to give a superior QOR and to trigger power-driven optimization. Power-driven optimization implies that the `opt_design` command now performs Power, Performance, and Area (PPA) trade-offs at the transform level at every step of timing optimization.

You can specify any one of the following options:

`none` : Specifies that no power optimization is to be performed.

low: Specify this option when power optimization is not the highest priority. Using the low effort level option indicates the following:

- Power optimization is performed only at certain stages of the `opt_design` flow, which is dependent on the value of the `opt_leakage_to_dynamic_ratio` attribute. When the `opt_leakage_to_dynamic_ratio` attribute is 1.0, `opt_design` will optimize leakage power after each timing optimization step and no high leakage cells are inserted during hold fixing. When `opt_leakage_to_dynamic_ratio` attribute is 0.0, `opt_design` will optimize dynamic power at the preCTS stage. For all values of `opt_leakage_to_dynamic_ratio` attribute between 0.0 and 1.0, both of the above are done.
- Timing optimization is power-aware but uses only low-effort techniques when choosing the best timing optimization fixes to implement.
- Timing closure remains the first priority and the worst negative slack achieved is the same as that in effort mode “ `none` ”.
- On an average, the CPU full flow runtime impact is +26% for a `opt_leakage_to_dynamic_ratio` of 0.5 as compared to that in effort mode “ `none` ”. This can vary depending on both the ratio being used and on the design.
- On an average, the MEM full flow impact is +5%.
- Utilization remains roughly at par with effort level “ `none` ” results.
- On an average, the overall standard cell dynamic power saving is -30% and can go up to -

69% as compared to that achieved in effort mode “none”. On an average, the overall standard cell leakage power saving is -41% and can go up to -90% on some designs. This is achieved with a opt_leakage_to_dynamic_ratio of 0.5 and will vary if this ratio is changed.

Note: The above numbers are estimates and are completely dependent on the type of design.Therefore, they may be very different from the averages given above.

high : Use the high effort level option when having low power is the highest priority. Using the high effort level option indicates the following:

- Power optimization is performed during all calls to opt_design , which is dependent on the value of the opt_leakage_to_dynamic_ratio attribute. Timing Optimization is power aware and uses high-effort techniques when choosing the best timing optimization fixes to implement.
- Timing closure should still be achievable, if it was previously, and in the majority of cases the same worst negative slack is reached as effort mode " none ".
- CPU full flow runtime impact is on an average +63% for a opt_leakage_to_dynamic_ratio of 0.5 compared to effort " none ". This can vary depending on both the ratio being used and on the design.
- MEM full flow impact is on an average +11%.
- Utilization is generally improved by -1.3% on average compared to effort level " none " results.
- On an average, the overall standard cell dynamic power saving is -32% and can go up to -70% as compared to that achieved in effort mode “none”. On an average, the overall standard cell leakage power saving is -54% and can go up to -98% on some designs. This is achieved with a opt_leakage_to_dynamic_ratio of 0.5 and will vary if this ratio is changed.

  Note : The above numbers are estimates and are completely dependent on the type of design.Therefore, they may be very different from the averages given above.

Related Command

- opt_design

opt_pre_route_ndr_aware

```
opt_pre_route_ndr_aware list
```

Default: "" (empty string)

Data_type: string, read/write

Enables NDR-based optimization by providing a list of NDR names. Using NDR-based optimization, GigaOpt assigns NDR on timing critical nets in order to improve timing. GigaOpt will automatically assess the list of NDRs provided and choose the one that can help timing. It is not recommended to use many NDRs from turn-around-time (TAT) point of view.

Related Command

- `opt_design`

opt_preserve_all_sequential

```
opt_preserve_all_sequential {true | false}
```

Default: `false`

Data_type: bool, read/write

Preserves sequential elements when netlists are optimized. When the `opt_remove_redundant_insts` attribute is set to `false`, unobservable or useless logic is removed. However, you can retain sequential elements by setting this attribute to `true`.

Related Command

- `opt_design`

opt_preserve_hpin_function

```
opt_preserve_hpin_function {true | false}
```

Default: false

Data_type: bool, read/write

Determines whether to preserve logical functions at hierarchical module ports. When set to false, only the hierarchical module ports that have a timing assertion applied on them will be preserved

(not deleted, duplicated, or polarity changed). This is the recommended setting since it allows more freedom for timing optimization while not impacting the netlist functionality or the timing constraints.

When set the true, every port of every hierarchical mode will be preserved (not deleted, duplicated, or polarity changed). This is very constraining.

Related Command

- `opt_design`

opt_remove_redundant_insts

```
opt_remove_redundant_insts {true | false}
```

Default: `true`

Data_type: bool, read/write

Controls whether or not timing optimization simplifies the netlist. The software recovers area, decreases congestion, and improves runtime by simplifying the netlist in the following ways:

- Removing dangling output instances
- Propagating constants
- Removing unobservable logic
- Remapping useless logic

The software respects the set_dont_touch constraint, so it does not remove the constrained flip-flops, and does not touch non-uniquified modules.

Note: This attribute is not enabled when the `design_flow_effort` attribute is set to `express`.

Note: If a netlist contains spare cells, ensure that you define them using the `set_spare_insts` command else they might be removed by this attribute.

Related Commands

- `opt_design`
- `set_spare_insts`

opt_resize_flip_flops

```
opt_resize_flip_flops {true | false}
```

Default: `true`

Data_type: bool, read/write

Controls whether timing optimization upsizes or downsizes flip-flops.

Related Command

- opt_design

opt_resize_level_shifter_and_iso_insts

```
opt_resize_level_shifter_and_iso_insts {true | false}
```

Default: `false`

Data_type: bool, read/write

Controls whether timing optimization resizes shifters and isolation instances. In default mode (false), shifters and isolations are considered "don't-touch" instances and are not changed during timing optimization. When this attribute is set to true, the shifters and isolation cells are resized for timing optimization by using opt_design.

Note: When this attribute is set to true, all the shifters and isolation cells can be resized even if they are marked as dont_touch.

Related Command

- opt_design

opt_resize_power_switch_insts

```
opt_resize_power_switch_insts {true | false}
```

Default: `false`

Data_type: bool, read/write

When set to `true`, enables resizing of power switch instances during optimization. By default, this

attribute is set to `false`, which means the power switch instances are not resized during optimization.

Related Command

- `opt_design`

opt_flop_pins_report

```
opt_flop_pins_report {true | false}
```

Default: `false`

Data_type: bool, read/write

Set this attribute to `true` for printing out full hierarchical pin names used in the incoming netlist for flops deleted during simplifyNetlist. The output file name format is `<userPrefix>_flops_pins.rpt`.

You can choose to write both files with pins or without pins (`opt_flops_report`) in same session .

Related Command

- `opt_design`

opt_flops_report

```
opt_flops_report {true | false}
```

Default: `false`

Data_type: bool, read/write

Set this attribute to `true` for printing full hierarchical flop names used in the incoming netlist for flops deleted during simplifyNetlist. The output file name format is `<userPrefix>_flops.rpt`.

You can choose to write both files with pins (`opt_flop_pins_report`) or without pins in same session.

Related Command

- `opt_design`

opt_sequential_genus_restructure_report_failure_reason

```
opt_sequential_genus_restructure_report_failure_reason {true | false}
```

Default: `false`

Data_type: bool, read/write

When set to true, the sequential critical region resynthesis (CRR) failure reasons are reported in the `SeqCrrFailReasons.rpt` file. The report includes the instance name and the corresponding failure reason.

Note: When CRR is enabled, Innovus uses Genus to restructure portions of the design to improve timing/area/power during setup optimization.

Related Command

- `opt_design`

opt_setup_target_slack

```
opt_setup_target_slack setuptargetslack
```

Default: `0.0`

Data_type: double, read/write

Specifies a target slack value in nanoseconds for setup analysis. During setup violation repair, the setup target slack is `opt_setup_target_slack` and the hold target slack is set to 0. Generally, you use both `opt_setup_target_slack` and `opt_hold_target_slack` for hold analysis. If you specify `opt_hold_target_slack` only, the `opt_setup_target_slack` value is 0.

Related Command

- `opt_design`

opt_target_based_opt_file

```
opt_target_based_opt_file  filename
```

Default: `"" (empty string)`

Data_type: string, read/write

Specifies the file containing information that target-based optimization (TBOpt) will use to do fine grain target-based optimization using the `opt_design -post_route -targeted` command.

The format of the TBOpt file is as follows:

```
NETS : {net1} {net2} {netN}
FROMTOS : {fromTerm1 toTerm1} {fromTerm2 toTerm2} {fromTermN toTermN}
INSTS : {inst1} {inst2} {instN}
PATHS : {endpoint1} {endpoint2} {endpointN}
END_POINT_ADJ : {endpoint1} {slack_adj: value in ps}
AREA_RECLAIM : {ModuleName} {x1 y1 x2 y2}
PWR_RECLAIM : {ModuleName} {x1 y1 x2 y2}
DRVS : {netName1} {netName2} {max_tran: e.g. 100}
```

An example of a TBOpt file is provided below:

```
NETS : {mod_i/mod_sub_i/reset_l_sync_3} {mod_i/mod_sub_i/FE_OCPN210_reset_l_sync_3}
INSTS : {mod_i/mod_sub_i/reset_l_sync_reg_15}
{mod_i/mod_sub_i/FE_OCPC210_reset_l_sync_16}
PATHS : {mod_i/mod_sub_i/ser_en/data_d1_reg_32/D}
```

Related Command

- opt_design

opt_target_based_opt_hold_file

```
opt_target_based_opt_hold_file hold_file_name
```

Default: `"" (empty string)`

Data_type: string, read/write

Specifies the file containing information that target-based optimization (TBOpt) will use to perform hold optimization.

```
set_db opt_target_based_opt_hold_file hold_file_name

opt_design -post_route -targeted -hold
```

Note: You can launch setup optimization followed by hold optimization using the following command:

```
opt_design -post_route -targeted -setup -hold
```

Hold optimization is only called in the `opt_target_based_opt_file_only` mode.

Related Command

- `opt_design`

opt_target_based_opt_file_only

```
opt_target_based_opt_file_only {true | false}
```

Default: true

Data_type: bool, read/write

This attribute is used along with the `opt_target_based_opt_file` attribute. It is set to `true` by default. When set to `true`, TBOpt will only optimize the nodes contained within the file specified using the `opt_target_based_opt_file` attribute. When set to `false`, the TBOpt will optimize other nodes in addition to those specified in the file, if the software considers them to be critical.

Related Command

- `opt_design`

opt_tied_inputs

```
opt_tied_inputs {true | false}
```

Default: `false`

Data_type: bool, read/write

When set to `true`, forces optimization of instances with inputs tied together, even if it worsens the objective gain - in terms of area, power, and delay.

Note: This attribute does not need to be set when the `eco_remove_tied_inputs` command is run. In that case this attribute is automatically set to `true` and instances with inputs tied together are optimized. After the command has been run, the attribute reverts to the value before executing the command.

Related Command

- opt_design

opt_time_design_compress_reports

```
opt_time_design_compress_reports {true | false}
```

Default: true

Data_type: bool, read/write

When set to `true`, this attribute forces `time_design` and `opt_design` to output timing reports in a compressed format.

Related Commands

- opt_design
- time_design

opt_time_design_expanded_view

```
opt_time_design_expanded_view {true | false}
```

Default: `false`

Data_type: bool, read/write

When set to true, this attribute will force `time_design` or `opt_design` to print summary timing reports as per the view in the log file.

Related Commands

- opt_design
- time_design

opt_time_design_num_paths

```
opt_time_design_num_paths number_of_paths
```

Default: `50`

Data_type: int, read/write

Allows you to select the number of paths that should be reported per path group during `time_design` or `opt_design` final summary reports.

Related Commands

- `opt_design`
- `time_design`

opt_time_design_report_net

```
opt_time_design_report_net {true | false}
```

Default: `true`

Data_type: bool, read/write

When set to `true`, it forces `time_design` or `opt_design` to print timing reports with the net section.

Related Commands

- `opt_design`
- `time_design`

opt_time_design_vertical_timing_summary

```
opt_time_design_vertical_timing_summary {true | false}
```

Default: `false`

Data_type: bool, read/write

When set to `true`, prints vertical timing summary after `time_design` and final summary report.

Related Commands

- opt_design
- time_design

opt_unfix_clock_insts

```
opt_unfix_clock_insts {true | false}
```

Default: true

Data_type: bool, read/write

Controls whether the software is allowed to resize fixed sequential elements and to move them during placement legalization. opt_design temporarily removes the fixed attribute of each sequential element at the beginning of this step and sets it back to its original value at the end. This attribute does not remove the routed wires of the clock tree. It minimizes the movement of sequential elements as much as possible. This attribute impacts only opt_design -post_cts and opt_design -post_route. You can set the following options:

- When set to true, the software is allowed to resize and move fixed registers.
- When set to false, the software is not allowed to resize or move fixed registers.

Note: If you specify true for this attribute, Cadence recommends you use the route_design command for routing your design.

Related Commands

- opt_design
- route_design

opt_skew

```
opt_skew {true | false}
```

Default: true

Data_type: bool, read/write

Controls whether timing optimization runs the opt_clock_skew command. This attribute produces

scheduling and latency files before CTS runs, or adds buffers and inverters after CTS, just as `opt_clock_skew` does. This attribute sets the `timing_analysis_useful_skew` attribute to true, so timing analysis can read the latency file. Useful skew optimization is run iteratively within `opt_design`, so you do not need to set this attribute more than once.

Note: This attribute is not enabled when the `design_flow_effort` attribute is set to `express`.

Note: This attribute is not supported during `opt_design -post_route`.

Related Commands

- `opt_design`
- `opt_clock_skew`

opt_skew_apply_delay_limits_to_full_flow

`opt_skew_apply_delay_limits_to_full_flow {true | false}`

Default: `false`

Data_type: bool, read/write

When set to `true`, specifies that the useful skew limit is to be applied to the whole flow. With this setting, the Early Clock Flow is modified so that pre-existing tool-generated useful skews count towards the limit imposed by the `opt_skew_max_allowed_delay` attribute. The tool-generated useful skew comprises any pin insertion delay (PID) of category useful_skew or global_skew. PIDs of other categories are not considered.

By default, this attribute is set to `false`, which means the pre-existing skews are not considered while calculating the limit imposed by using the `opt_skew_max_allowed_delay` attribute.

opt_skew_ccopt

`opt_skew_ccopt {none | standard | extreme}`

Default : `standard`

Data_type: enum, read/write

Specifies the effort for useful skew in both CTS and post-CTS flow. It replaces the attributes, `opt_skew_pre_cts, opt_skew_post_cts .`

This attribute setting is only respected by `ccopt_design` and `opt_design -post_cts`.

- `none`: low effort `ccopt_design` with no skewing
- `standard`: low effort `ccopt_design`. This is the default setting.
- `extreme`: high effort `ccopt_design`

Note: To achieve the best possible timing result, the `extreme` setting will significantly increase the execution time of the `ccopt_design` command. Also, for best results the extreme setting should be used with `set_db design_flow_effort extreme` attribute setting.

Related Commands

- `opt_design`
- `opt_clock_skew`

opt_skew_post_route

```
opt_skew_post_route {true | false}
```

Default : `true`

Data_type: bool, read/write

When set to `false` , it disables useful skew in post-Route optimization.

Related Commands

- `opt_design`
- `opt_clock_skew`

opt_skew_pre_cts

```
opt_skew_pre_cts {true | false}
```

Default : `true`

Data_type: bool, read/write

When set to `false` , it disables useful skew in preCTS optimization.

Related Commands

- `opt_design`
- `opt_clock_skew`

opt_skew_delay_pre_cts

```
opt_skew_delay_pre_cts {true | false}
```

Default: true

Data_type: bool, read/write

Specifies that the useful skew feature is allowed to delay sequential elements in preCTS stage, in addition to just advancing them. This enables a more aggressive algorithm which is recommended when timing closure is gated by optimal skewing.

Related Commands

- `opt_design`
- `opt_clock_skew`

opt_skew_macro_only

```
opt_skew_macro_only {true | false}
```

Default: false

Data_type: bool, read/write

Allows you to limit skewing only for macro cells. When set to true, regular sequential elements (such as flip flops) will not be skewed.

Related Commands

- `opt_design`
- `opt_clock_skew`

opt_skew_max_allowed_delay

```
opt_skew_max_allowed_delay delay
```

Default: 1.0ns

Data_type: double, read/write

Limits the amount of slack (in nanoseconds) that the software can borrow from neighboring flip-flops when performing useful skew operations. The delay calculation and RC extraction methods might differ from those of sign-off tools, so other setup violations might occur if the software borrows too much slack. By having control over slack borrowing, you can prevent these setup violations. Limiting borrowed skew also limits the clock tree skew to avoid large hold violations.

Value Range: `min=0 max=1000`

Note:

This attribute is not honored in the `post_cts` mode of `opt_clock_skew`.

Note: In the non-early clock flow, when `opt_clock_skew` is triggered inside opt_design, and this attribute is set to its default value, the default value of this attribute will change from 1 to become a certain percentage of clock cycle. This is determined by the software on the basis of internal calculations.

Related Commands

- `opt_design`
- `opt_clock_skew`

opt_skew_min_allowed_delay

```
opt_skew_min_allowed_delay delay
```

Default: `0.0`

Data_type: double, read/write

Allows the tool to skip any skewing that would be smaller than the minimum delay that is set. The purpose is to avoid creating very small delay macro models which would not be correctly honored during CTS stage but also to reduce the amount of macro models generated.

Value Range: `min=0 max=1000`

Note: This attribute is supported in both standalone `opt_clock_skew` command and for

`opt_useful_skew` command called within `opt_design`. This is called by setting the `opt_skew` attribute to true.

Related Commands

- `opt_design`
- `opt_clock_skew`

opt_skew_no_boundary

```
opt_skew_no_boundary {true | false}
```

Default: false

Data_type: bool, read/write

Excludes or includes boundary sequential cells in addition to ordinary sequential elements in useful skew calculations.

Related Commands

- `opt_design`
- `opt_clock_skew`

opt_verbose

```
opt_verbose {true | false}
```

Default: `false`

Data_type: bool, read/write

Controls the amount of information timing optimization outputs to the log file. Specify true to output additional information to use to debug design issues. When you set this attribute to true, the various reasons for not fixing hold violations are listed in the log file.

For more information, see the section below.

Reasons for Hold Violations to Remain Unfixed

The following table lists the various reasons for hold violations to remain unfixed:

Reason	Description
No legal location	There are no valid locations to insert buffers on those violated nets. Limit the local placement congestion and use instance/cell padding during initial placement. In parallel, to allow the tool to insert buffers at non-legal location, set the following attribute to true: `set_db opt_hold_allow_overlap true`
Degradation in setup register-to-register WNS	The nets are violated in hold but they are also a part of the register-to-register setup timing-critical paths. So inserting a buffer would degrade the setup register-to-register WNS. To fix these nets at the cost of register-to-register setup degradation, set a negative setup target slack using the `opt_setup_target_slack` attribute.
Degradation in setup TNS	The nets are violated in hold but they are also timing critical during setup. So, inserting a buffer would increase the setup TNS. To fix these nets at the cost of setup TNS degradation, set the following attribute to true: `set_db opt_hold_allow_setup_tns_degradation true`
Degradation in setup WNS	The nets are violated in hold but they are also setup timing critical. So inserting a buffer would degrade the setup WNS. To fix these nets at the cost of setup degradation, set a negative setup target slack using the `opt_setup_target_slack` attribute.
Degradation in hold	The nets are violated in hold but inserting a buffer would further degrade hold timing. Check the topology of those nets and probably enlarge the list of available buffer/delay cells that are allowed to be used.

Degradation in maximum capacitance violation	The nets are violated in hold but inserting a buffer would create a new setup maximum capacitance violation. To fix these nets at the cost of DRV degradation, set the following attribute: `set_db opt_drv false`
Degradation in maximum transition violation	The nets are violated in hold but inserting a buffer would create a new setup maximum transition violation. To fix these nets at the cost of DRV degradation, set the following attribute: `set_db opt_drv false`
DRV violations	The nets are violated in hold but also in setup DRV, so they are skipped. To fix these nets at the cost of DRV degradation, set the following attribute: `set_db opt_drv false`
Nets are set as `dont_touch`	To allow hold fixing on these nets, remove the `dont_touch` attribute.
Nets are routed incorrectly	Verify the routing wires for those nets.
Nets are clock nets	No buffers can be inserted on clock nets. You should check your timing constraint file.
None of the always-on-buffers are usable	Verify the environment to allow the use of always-on-buffers.
None of the valid mode is found on a net	The tool is not able to insert a buffer on such a net. This could be related to specific design cases.
Any internal failure	The tool was not able to find a solution for fixing the violation of those nets. This could be related to libraries or specific design cases.

Related Command

- `opt_design`

opt Category Object Attributes

The following categories of object attributes are available in the opt category:

Object Category Name	Attributes
inst	dont_touch dont_touch_sourcesdont_touch_effective
hinst	dont_touch dont_touch_sourcesdont_touch_effective read_only_effective dont_use_cells use_cells dont_use_cells_effective dont_touch_hports is_ilm
module	dont_touch dont_use_cellsuse_cells dont_use_cells_effective dont_touch_hports is_ilm
design	read_only dont_touchdont_touch_sources dont_use_cells use_cells dont_use_cells_effective
partition	read_only
net	dont_touch
hnet	dont_touch
pin	dont_touch
hpin	dont_touch

base_cell	dont_touch dont_use

To view help for any object type attribute across different object types, type the following command:

```
help attributeName
```

For example, for help on `dont_touch`, `dont_touch_effective`, and `dont_touch_source` for all objects type this:

```
help dont_touch*
```

For help on one object attribute, use this:

```
help inst dont_touch
```

read_footprint

```
read_footprint
[-help]
file_name
```

Loads a footprint file to update the footprint information, which is obtained from the timing library file during design import. Loading a footprint file takes precedence over what is currently defined in the Stylus session. That is, after a footprint file is loaded for a session, only the cells in the footprint file are considered. This feature lets you run individual timing optimization commands with a limited set of cell footprints.

When the command is run successfully, the software gives the following message:

```
----------------------------------------
Clearing footprints for all libraries
Loading footprints for all corners.
----------------------------------------
```

Parameters

`-help`	Outputs a brief description that includes type and default information for each `read_footprint` parameter. For a detailed description of the command and all of its parameters, use the man command: `man read_footprint`.
`file_name`	Specifies the name of the footprint file. *Data_type*: string, required

Examples

- The following command loads the footprint file `INV.fpt`:

  ```
  read_footprint INV.fpt
  ```

  After loading the footprint file, the software ignores the previous footprint information.

- When the `read_footprint` command reads the output from the `report_footprints` command, the `read_footprint` command ignores all other fields except the library name, cell name, and the footprint name. The footprint file format for the `read_footprint` command is as follows:

```
Library:library_name
cell_name footprint_name
cell_name footprint_name
```

For example:

```
Library:stdcellLib
xr03d1 xr03_5
xr03d2 xr03_5
```

Related Information

- check_footprint
- report_footprints
- set_buffer_footprint
- set_delay_footprint
- set_inverter_footprint
- The following section in Optimizing Timing chapter of *Innovus Stylus Common UI User Guide*
  - Using Cell Footprints

report_always_on_buffer

```
report_always_on_buffer
[-help]
[-all]
[-power_domain power_domain_name]
[-verbose]
```

Reports the available always-on buffers and inverters per domain.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `report_always_on_buffer` parameter. For a detailed description of the command and all of its parameters, use the man command: `man report_always_on_buffer`
`-all`	Reports the always-on buffers and inverters for all domains. If you specify no parameter, even then it reports the always-on buffers and inverters for all domains. *Data_type*: bool, optional
`-power_domain` *power_domain_name*	Reports the always-on buffers and inverters for the specified domain. **Note**: Parameters `-all` and `-power_domain` are mutually exclusive. *Data_type*: string, optional
`-verbose`	Reports the possible reason why a domain does not have always-on buffers and inverters. It can also report the always-on cell list per domain if the always-on cells are available for the domain. *Data_type*: bool, optional

Examples

- The following command reports the always-on buffers and inverters for the `PD1` domain:

  ```
  report_always_on_buffer -power_domain PD1
  ```

  The software displays the following message:

  ```
  PowerDomain "PD1" (pd tag = "1") has 2 always on buffer(s) to use
  ```

Related Information

- `opt_design`
- opt Category Attributes chapter of *Innovus Stylus Common UI Text Command Reference*
- Optimizing Timing chapter of *Innovus Stylus Common UI User Guide*

report_critical_nets

```
report_critical_nets
[-help]
[-max_slack double]
[-number integer]
-out_file fileName
[-view viewName]
```

Generates a file containing a list of nets which have critical slack for the currently specified (setup or hold) timing analysis mode.

Parameters

`-help`	Prints a brief description that includes type and default information for each `report_critical_nets` parameter. For a detailed description of the command and all of its parameters, use the man command: `man report_critical_nets`
`-max_slack double`	Reports nets with slack less than the specified slack value. *Data_type*: float, optional
`-number integer`	Restricts the number of critical nets reported to the first *integer* most critical nets. *Data_type*: int, optional
`-out_file fileName`	Specifies the name of report file. *Data_type*: string, optional
`-view viewName`	Generates the report based on the specified analysis view only. You can specify this parameter only when the software is in multi-mode multi-corner timing analysis mode. *Data_type*: string, optional

Command Order

Use this command after the timing graph is built.

Example

The following command generates a report named `critnet` that lists critical nets:

```
report_critical_nets -out_file critnet
```

Related Information

- `opt_design`
- `set_db`
- `time_design`
- opt Category Attributes

report_critical_insts

```
report_critical_insts
[-help]
[-cost_type {worst_slack | total_slack | path_count | start_end_point_count}]
[-estimate]
[-include_sequential]
[-max_insts value]
[-max_slack slack]
[-out_file file_name]
[> file]
[>> file]
```

Reports instances in critical paths where the slack is less than the target slack.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `report_critical_insts` parameter. For a detailed description of the command and all of its parameters, use the man command: `man report_critical_insts`.

`-cost_type`	Specifies the criteria for choosing the critical instance. You can use the `-max_slack` and `-cost_type` parameters together to filter the paths to only those with slack worse than the specified *`slack`* value, then report the filtered set of paths based on the specified cost. *Default:* `worst_slack` Select one of the following options: `worst_slack`: Sets the criteria as the worst slack of all points that pass through an instance. `total_slack`: Sets the criteria as a sum of the slack of the end points. `path_count`: Sets the criteria as the sum of the total paths through the instance. `start_end_point_count`: Sets the criteria as the sum of the total start and end points connected to the instance. *Data_type*: string, optional
`-estimate`	Generates a critical instance report based on the fast algorithm (linear-complexity) when cost type = `total_slack` to give a slightly pessimistic estimation that may include some false paths. Use this parameter in early design phase. *Default:* `false` *Data_type*: bool, optional
`file`	Redirects report to the outfile. If the file already exists, the software overwrites it. **Note:** The file name must be the last argument in the list. *Default:* `stdout` *Data_type*: redirect_operator, optional
`-include_sequential`	Includes the sequential instances group in the output. *Default:* false (Reports combinatorial instances only.) *Data_type*: bool, optional

`-max_insts` *value*	Specifies the number of worst cost instances to report. *Default:* `20` *Data_type*: int, optional
`-max_slack` *slack*	Specifies the lower bound of the slack range. Only paths with slack worse than the specified *slack* value are reported. You can use the `-max_slack` and `-cost_type` parameters together to filter the paths to only those with slack worse than the specified *slack* value, then report the filtered set of paths based on the specified cost. *Default:* `0` *Data_type*: float, optional
`-out_file` *file_name*	Specifies the report output file. *Default:* `stdout` *Data_type*: string, optional

Command Order

Use this command after the timing graph is built by `time_design/opt_design`.

Examples

The following command reports the critical instance information based on the worst slack of all points that pass through an instance:

```
report_critical_insts -cost_type worst_slack
# ----------------------------------------
# Inst:combinatorial  Cell        Cost
# ----------------------------------------
I3                    BUFX2       -0.336
I10                   AND2X1      -0.336
I1                    BUFX2       -0.117
I2                    BUFX2       -0.117
----------------------------------------
```

The following command reports the critical instance information based on the worst slack of all points that pass through an instance, for paths with slack worse than -0.2:

```
report_critical_insts -cost_type worst_slack -max_slack -0.2
# ------------------------------------------------
# Inst:combinatorial    Cell          Cost
# ------------------------------------------------
I3                      BUFX2       -0.336
I10                     AND2X1      -0.336
------------------------------------------------
```

The following command reports the critical instance information based on the sum of the slack of the end points:

```
report_critical_insts -cost_type total_slack
# ------------------------------------------------------
# Inst:combinatorial    Cell       Cost    Start  End
# ------------------------------------------------------
I3                      BUFX2     -0.336    1     1
I10                     AND2X1    -0.336    2     1
I1                      BUFX2     -0.117    0     1
I2                      BUFX2     -0.117    0     1
------------------------------------------------------
```

The following command reports the critical instance information based on the sum of the total paths through the instance:

```
report_critical_insts -cost_type path_count
# ------------------------------------------------
# Inst:combinatorial   Cell       Cost    Start  End
# ------------------------------------------------
I10                    AND2X1      4       2     1
I3                     BUFX2       2       1     1
I1                     BUFX2       2       0     1
I2                     BUFX2       2       0     1
------------------------------------------------
```

The following command reports the critical instance information based on the sum of the total start and end points connected to the instance:

```
report_critical_insts -cost_type start_end_point_count
# ------------------------------------------------------
# Inst:combinatorial    Cell       Cost    Start  End
# ------------------------------------------------------
I10                      AND2X1      2       1     1
I3                       BUFX2       2       1     1
I1                       BUFX2       2       1     1
I2                       BUFX2       2       1     1
------------------------------------------------------
```

The following command reports the critical instance information based on the sum of the total start and end points connected to the instance, for paths with slack worse than -0.2:

```
report_critical_insts -cost_type start_end_point_count -max_slew -0.2
# ------------------------------------------------------
# Inst:combinatorial    Cell       Cost    Start  End
# ------------------------------------------------------
I10                     AND2X1      2       1     1
I3                      BUFX2       2       1     1
------------------------------------------------------
```

The following command generates an output file named `inst.rpt`, which includes all the instance names and cell names. The instances are on the critical paths whose slack is less than 0 ns.

```
report_critical_insts -out_file inst.rpt
```

The following command generates an output file named `inst.rpt2`, which includes all the instance names and the cell names. The instances are on critical paths whose slack is worse than `-0.3` ns.

```
report_critical_insts -out_file inst.rpt2 -max_slack -0.3
```

Related Information

- `opt_design`
- `set_db`
- `time_design`
- opt Category Attributes

report_critical_pins

```
report_critical_pins
[-help]
-out_file file_name
```

Performs slack analysis and writes the names of all critical terminals to the specified file. As a result, deterioration of setup slack can be avoided while the Innovus software repairs hold time violations.

Parameters

`-help`	Prints a brief description that includes type and default information for each `report_critical_pins` parameter. For a detailed description of the command and all of its parameters, use the man command: `man report_critical_pins`
`-out_file file_name`	Specifies the name of report file. *Data_type*: string, required

Command Order

Use this command after the timing graph is built by `time_design/opt_design`.

Example

- The following attribute and command write the setup critical terminals into a file called `critTerm`:

  ```
  set_db timing_analysis_check_type setup
  report_critical_pins -out_file critTerm
  ```

Related Information

- `opt_design`
- `set_db`
- `time_design`
- opt Category Attributes

report_footprints

```
report_footprints
[-help]
[-dont_touch_and_use]
[footprint_cell]
-out_file file_name
```

Creates a file that contains footprint_cell attribute information from the timing library. Use this file to check your design for missing footprint_cell function names and timing information. You can edit the output file and use it to load the correct footprint_cell information.

Parameters

`-help`	Prints a brief description that includes type and default information for each `report_footprints` parameter. For a detailed description of the command and all of its parameters, use the man command: `man report_footprints`
`-dont_touch_and_use`	When this parameter is specified, cell footprint_cells are reported for all cells, including those cells that have dont_touch or dont_use attribute. When this parameter is not specified, cell footprint_cells are reported for only those cells that do not have dont_touch and dont_use attributes. *Data_type*: bool, optional
footprint_cell	Specifies the footprint_cell to report. If you do not specify this parameter, reports all footprint_cells. *Data_type*: string, optional
`-out_file` *file_name*	Specifies the footprint_cell report file. *Data_type*: string, optional

Command Order

Use this command after importing the design or after using the `read_footprint` command.

Examples

- The following command creates a footprint_cell report file named `INV.rpt`:

  ```
  report_footprints -out_file INV.rpt
  ```

- The following output from the `report_footprint_cells` command shows the format of the footprint_cell file:

  ```
  # Format:
  # Library:<library name>
  # cell footprint_cell drvResR drvResF functions

  # footPrint: INV nrCell: 1 Library: dp_ram128x32_4x1
  Library:stdCellLib
  inv_2 INV     1            110.983    89.427    x=(~a)
  ```

  Lines beginning with a # are comment lines, which are ignored when the footprint_cell file is loaded. The header shows the meaning of each field:

  - `Library: stdCellLib` (with no spaces in the string) is the library where the footprint_cell is defined.
  - `inv_2` is a footprint_cell cell.
  - `INV` is the footprint_cell name.
  - `1` is the number of functions.
  - `110.983` is the average rise driving resistance.
  - `89.427` is the average fall driving resistance.
  - `x=(~a)` is the output function of the cell.

Related Information

- `check_footprint`
- `read_footprint`
- `set_buffer_footprint`
- `set_delay_footprint`
- `set_inverter_footprint`
- The following section in Optimizing Timing chapter of *Innovus Stylus Common UI User Guide*
  - Using Cell Footprints

report_ignored_nets

```
report_ignored_nets
[-help]
-out_file file_name
```

Generates a report that lists the nets that are ignored by timing optimization (`opt_design`) or that might impact timing closure. You can use this command after loading a design with timing constraints.

The report lists the following types of nets:

- Nets that are ignored by timing optimization
- Nets that might impact timing closure but are not ignored

Nets that are ignored by timing optimization are listed below:

`dontTouchNets`	Nets set as don't touch by the `set_dont_touch` SDC constraint
`externalNets`	Nets connected to I/O pad or area I/O instances
`topLevelPotentialTriStateNets`	Nets connecting tristate drivers to top-level I/Os
`clockNets`	Nets marked `clock` by the timing engine
`specialNets`	Nets that have the attribute `specialnet`
`pwrGndNets`	Nets that have the attribute `power/ground`
`analogNets`	Nets that have the attribute `analog`
`noDriverNets`	Nets that do not have a driver
`Fixed Wires/Cover Wires`	Nets that have a `Fixed/Cover` wire attribute

Nets that might impact timing closure but are not ignored are listed below:

`multiDriverNets`	Nets that have multiple drivers

Parameters

`-help`	Prints a brief description that includes type and default information for each `report_ignored_nets` parameter. For a detailed description of the command and all of its parameters, use the man command: `man report_ignored_nets`
`-out_file` *`file_name`*	Specifies a file name for the report. *Data_type*: string, required

Related Information

- `opt_design`
- opt Category Attributes

report_lef_equivalent_cells

```
report_lef_equivalent_cells
[-help]
-lib_cell cell_name
[-out_file file_name]
```

Provides support to debug certain cells that may be non-equivalent without detailed analysis of the LEF. For example, if two cells have the same function and size they still may not be LEF equivalent.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `report_lef_equivalent_cells` parameter. For a detailed description of the command and all of its parameters, use the man command: `man report_lef_equivalent_cells`
`-lib_cell` *`cell_name`*	Specifies the cell names. You can also use wildcards or specify individual cells. *Data_type*: string, required
`-out_file` *`file_name`*	Specifies the output file name for reporting LEF equivalent cells. All the equivalent and non-equivalent cells are saved in the output file. *Data_type*: string, optional

Examples

- The following example command reports equivalence information on all cells and outputs to a file:

  ```
  report_lef_equivalent_cells -lib_cell * -out_file equivalency_file.txt
  ```

- To report equivalence information on a specific cell, say `SCBUF1X`, you can use the following command:

  ```
  report_lef_equivalent_cells -lib_cell SCBUF1X
  ```

- To report equivalence information on all cells that begin with `SCBUF`, you can use the following

command:

```
report_lef_equivalent_cells -lib_cell SCBUF*
```

report_length_violations

```
report_length_violations
[-help]
[-all | -no_global_nets]
[-excluded_nets_file exc_net_file_name]
[-histogram]
[-included_nets_file sel_net_file_name]
[-out_file file_name]
[-significant_digits integer]
```

Runs timing analysis and reports nets that exceed the maximum length constraint set by `opt_max_length` attribute.

Parameters

`-help`	Prints a brief description that includes type and default information for each `report_length_violations` parameter. For a detailed description of the command and all of its parameters, use the man command: `man report_length_violations`
`-all`	Reports violations on all nets. *Data_type*: bool, optional
`-excluded_nets_file` *exc_net_file_name*	Specifies a file that lists the names of hierarchical nets that are to be excluded from the report. *Data_type*: string, optional
`-histogram`	Generates a histogram in addition to the default reporting. *Data_type*: bool, optional
`-included_nets_file` *sel_net_file_name*	Specifies a file that lists the names of hierarchical nets to consider in the report. Only violations on the nets listed in the file are considered in the report. **Note**: The parameters, `-included_nets_file` and `-excluded_nets_file` are not mutually exclusive. If both parameters are specified, the `-excluded_nets_file` parameter overides the `-included_nets_file` parameter. *Data_type*: string, optional
`-no_global_nets`	Reports violations on all nets except for power, ground, and clock nets. **Note**: The parameters, `-all` and `-no_global_nets` are mutually exclusive. *Data_type*: bool, optional
`-out_file` *file_name*	Specifies the report file name. *Data_type*: string, optional
`-significant_digits` *integer*	Allows you to specify the number of significant digits. *Default*: 3 *Data_type*: int, optional

Related Information

- `opt_design`
- The following opt Category Attribute:
    - `opt_max_length`

report_multibit

```
report_multibit
[-help]
[-out_file file_name]
[-cell cell_name | -library | -reason_not_merged unmerged_reason | -not_merged_summary
| -equiv]
[-cell cell_name | -type {comb | latch | flop | seq | iso | ls}]
[-hinst hinst_name | -library | -cell cell_name | -equiv ]
```

Reports all the multi-bit flip-flops that have been identified during loading the design. Use this command after all the libraries are loaded. When this command is specified with no parameters, a default summary report is printed. By default, the seqential cells will be reported. Both, the multi-bit percentages merged, and a summary of why flops were not merged is printed. This is equivalent to the `report_multi_bit_ffs -statistics` setting, with the addition of unused bits.

There are three basic types of multibit: sequential, combo, and isolation/level shifter. This command supports specifying the type of multibit to report. The `-type` parameter is a global filter for all other options.

The default option is `seq`. If the `-type` option is not specified then reporting will take place for sequential cells.

A set of options separated by a "`|`" in one "`[ ]`" listed below cannot be used together and the software will report an error in that case. Although options from different "`[ ]`" can be coupled together. The sets of mutually exclusive options are provided below.

```
[-cell <cell name> | -library | -reason_not_merged <unmerged reason> | -
not_merged_summary | -equiv ]
```

```
[-cell <cell name> | -type <type>]
```

```
[-hinst <hinst> | -library | -cell <cell name> | -equiv ]
```

Parameters

`-help`	Prints a brief description that includes type and default information for each `report_multibit` parameter. For a detailed description of the command and all of its parameters, use the man command: `man report_multibit`.

`-cell` *cell_name*	Reports the cell details and equivalence information of a single library cell.
`-equiv`	Reports equivalence summary of the library cells of the selected type.
`-hinst` *hinst_name*	Specifies the hinst on which the reporting is to be done. The default is the design.
`-library`	Reports the flip-flop/latch equivalence. This parameter is similar to the `report_multi_bit_ffs -all -info/-equivalence` setting. This parameter honors the `-type` parameter setting. *Data_type*: bool, optional
`-not_merged_summary`	When specified, reports the summary of unmerged instances. *Data_type*: bool, optional
`-out_file` *file_name*	Specifies the name of the output file. *Data_type*: string, optional
`-reason_not_merged {partition cell_equivalent constraint scan_connection location timing power user legalization bus all}`	
	Reports the instances that could not be merged for the specified reason. Specify any of the reasons in the argument or specify `all` to report all instances along with the reason why they could not be merged. `partition`: odd number of flops in one partition `cell_equivalent`: no usable equivalent multi-bit flip-flop cell `constraint`: Constraints by sdc, rule, or user settings `scan_connection`: no scan connection to other flops in the same group `location`: too far away with other flops in the same group `timing`: exceed critical path threshold or MBFF merging that degrades timing `power`: all available multi-bit cells causing power degradation `user`: user set to be excluded in flop merge `bus`: non-bus flops `all`: all the reasons of not merged *Data_type*: enum, optional

`-type {comb | latch | flop | seq | iso | ls}`	Reports for the specified type of cells. By default, only sequentials are reported. You can specify anyone of the following reports: `comb`: reports all multibit cells implementing a combinational function. For example, a cell implementing 3 ANDs – so, 6 inputs and 3 outputs `latch`: reports multi-bit latches in the design. `flop`: reports multi-bit flops. `seq`: reports all sequential cells. isolation/level shifter (`iso/ls`): reports multi-bit cells implementing a low power function. For example, as 8-bit level-shifter cell. *Default*: `seq` *Data_type*: enum, optional

Examples

Use the following command to print a default summary report. By default, the seqential cells will be reported.

```
report_multibit
```

The software returns the following information:

```
================================================================================
Current Design Sequential Cells Statistics
================================================================================
+--------------------------------+----------+---------------+------------+
|                                | Latches  | Flops         | All Seq Cells |
+--------------------------------+----------+---------------+------------+
| Single-Bit Count               | 1157     | 101           | 1258          |
| Multi-Bit Count                | 0        | 2657          | 2657          |
|  2-Bit Count (Used-Bit %)      | 0        | 29(100.00)    | 29 (100.00)   |
|  4-Bit Count (Used-Bit %)      | 0        | 2628(100.00)  | 2628 (100.00) |
| Total Bit Count                | 1157     | 10671         | 11828         |
|  -Merged                       | 0        | 10570         | 10570         |
|  -Un-Merged                    | 1157     | 101           | 1258          |
| Total Cell Count               | 1157     | 2758          | 3915          |
| Total Cells With Unused Bits   | 0        | 0             | 0             |
| Total Unused Bits Count (%)    | 0 (0.00) | 0 (0.00)      | 0 (0.00)      |
| Bits per cell                  | 1.00     | 3.87          | 3.02          |
| Total Clock Pin Cap            | 675.69   | 1602.71       | 2278.40       |
```

```
| Multibit Conversion %          | 0.00    | 99.05       | 89.36         |
+--------------------------------+---------+-------------+---------------+
=============================================================
Sequential Multibit Cells Usage Statistics
=============================================================
+--------------------------+---------+-------+---------------+
|                          | Latches | Flops | All Seq Cells |
+--------------------------+---------+-------+---------------+
| Usable 2-bit cell        | 0       | 6     | 6             |
| Dont-use/touch 2-bit cell | 0       | 0     | 0             |
| Usable 4-bit cell        | 0       | 6     | 6             |
| Dont-use/touch 4-bit cell | 0       | 0     | 0             |
+--------------------------+---------+-------+---------------+

=================================================
Sequential Multibit cells usage statistics
=================================================
+-----------------------+------+----------+-------+
| Seq_Mbit libcell      | FF/L | Bitwidth | Count |
+-----------------------+------+----------+-------+
| HD40DG_LH_2SDFPQX4_P4  | FF   | 2        | 3     |
| HD40DG_LH_2SDFPRQX4_P4 | FF   | 2        | 26    |
| HD40QG_LH_4SDFPQX4_P4  | FF   | 4        | 2325  |
| HD40QG_LH_4SDFPRQX4_P4 | FF   | 4        | 303   |
+-----------------------+------+----------+-------+
| Total                 |      |          |  2657 |

=============================================================================
Rejection Reason Summary
=============================================================================
+-----------------+------------+-----------------------------------------------+
| Category        | Rej. Count | Reason |
+-----------------+------------+-----------------------------------------------+
| Partition       | 19         | Odd number of flops in one partition          |
| Cell Equivalent | 7          | No usable equivalent multi-bit flip-flop cell
                               (check 'Ignored single-bit flop-flop'
                               in the 'Pre-merge design statistics')         |
| Constraint      | 71         | Constraints by sdc,rule or user settings
                               (check 'Nonremovable single-bit flip-flop'
                               in the 'Pre-merge design statistics')
+-----------------+------------+-----------------------------------------------+
```

Use the following command to print a default summary report. By default, the sequential cells will be reported..

```
report_multibit -reason_not_merged Partition
```

The software returns the following information:

```
Reason of not merging:     Partition    Odd number of flops in one partition
Unmerged Sequential instances (19):
-----------------------------------
ethreg1/MIIADDRESS_1/DataOut_reg[4]
ethreg1/MIICOMMAND0/DataOut_reg[0]
ethreg1/MIICOMMAND1/DataOut_reg[0]
ethreg1/MIICOMMAND2/DataOut_reg[0]
ethreg1/SetTxCIrq_reg
ethreg1/TXCTRL_2/DataOut_reg[0]
maccontrol1/receivecontrol1/ByteCnt_reg[4]
maccontrol1/receivecontrol1/Divider2_reg
macstatus1/DeferLatched_reg
macstatus1/ReceiveEnd_reg
miim1/shftrg/LinkFail_reg
rxethmac1/CrcHashGood_reg
rxethmac1/rxcounters1/DlyCrcCnt_reg[0]
rxethmac1/rxcounters1/IFGCounter_reg[4]
txethmac1/txstatem1/StateJam_q_reg
wishbone/RxBDReady_reg
wishbone/TxPointerRead_reg
wishbone/rx_fifo/cnt_reg[4]
wishbone/tx_fifo/cnt_reg[4]
```

Related Information

- eco_clone_flop
- eco_split_complex_flop
- report_multi_bit_ffs
- write_multi_bit_flop_mapping_file
- The following opt Category Attributes:
  - opt_multi_bit_flop_opt
  - opt_allow_multi_bit_on_flop_with_sdc
- The following sections in Optimizing Timing chapter of *Innovus Stylus Common UI User*

Guide

- Reporting the MBFFs

report_multi_bit_ffs

```
report_multi_bit_ffs
[-help]
[-flops]
[-latches]
[[[-info | -equivalence | -usage] -all] | -cell cell name | -statistics | -
not_merged_inst_list | -reason_not_merged unmerged_reason]
[-out_file file_name]
```

Reports all the multi-bit flip-flops that have been identified during loading the design. Use this command after all the libraries are loaded. You can choose to report information about either all the multi-bit flip-flops or one multi-bit flip-flop cell by specifying the name of the cell. The reported information can be printed on the console and written to a file.

Parameters

`-help`	Prints a brief description that includes type and default information for each `report_multi_bit_ffs` parameter. For a detailed description of the command and all of its parameters, use the man command: `man report_multi_bit_ffs`.
`-all`	Reports all the multi-bit flip-flops in the design. The parameters `-all` and `-cell` are mutually exclusive. If neither `-all` nor `-cell` is specified, the command considers -`all` by default and reports all the MBFFs in the design. The parameters, `-equivalence`, `-info`, and `-usage` are used along with the `-all` parameter. However, If none of these parameters are specified, the command will report all the information reported by the `-info` , `-equivalence` , and `-usage` parameters. *Default*: `report_multi_bit_ffs -all` is used for `report_multi_bit_ffs`. **Note**: When the `opt_multi_bit_flop_reorder_bits` attribute is set to `true`, `power`, or `timing`, a "Bit-strength (strong to weak)" column will also be added in the report. *Data_type*: bool, optional

`-cell` *cell_name*	Specifies the name of the flip-flop for which information is to be reported. This parameter reports information, equivalence, and usage for the specified flip-flop. You can specify both single-bit and multi-bit flip flops for reporting this information. **Note**: When the `opt_multi_bit_flop_reorder_bits` attribute is set to `true`, `power`, or `timing`, a "Bit-strength (strong to weak)" column will also be added in the report. *Data_type*: string, optional
`-equivalence`	Reports multi-bit flip-flop equivalence. This parameter is used along with the `-all` parameter. It reports all the equivalent multi-bit flip-flop cells and corresponding single-bit flip-flop cells in the design. The cells are reported in groups and ordered on size. **Note**: The parameters, `-equivalence`, `-info`, and `-usage` are mutually exclusive. *Data_type*: bool, optional

`-flops`	When this parameter is specified, the software will report information about the flops. When no parameter is specified, `report_multi_bit_ffs` command prints the information for both, latches and flops. This parameter can be specified along with the following parameters: `report_multi_bit_ffs -flops -all`: Reports all multi-bit flops in the design. `report_multi_bit_ffs -flops -equivalence`: Reports multi-bit flops equivalence. `report_multi_bit_ffs -flops -usage`: Reports all multi-bit flops usage. `report_multi_bit_ffs -flops -info`: Used along with `-all`. This parameter reports the properties of all the multi-bit flops in the design. For example, `Name`, `NumBits`, `Don't Use/ Don't Touch`, `Dimensions`, `ScanType`, `Stateless Leakage`, `Stateless Dynamic`, and `LibName`. `report_multi_bit_ffs -flops -cell`: Specifies the name of the flop for which information is reported. `report_multi_bit_ffs -flops -statistics`: Reports the single-bit and multi-bit flop count in the current design.
`-info`	This parameter is used along with the `-all` parameter. This parameter reports the following properties of all the multi-bit flip-flop cells in the design: `Name`, `NumBits`, `Don't Use/ Don't Touch`, `Dimensions`, `ScanType`, `Stateless Leakage`, `Stateless Dynamic`, and `LibName`. **Note**: The unit of the numbers reported in the "Dimensions" column is in terms of DB-unit (`dbu`). To convert it to `um`, use the following conversion: `set dimUm [expr $dimInDbU / [uu2dbu 1]]` **Note**: When the `opt_multi_bit_flop_reorder_bits` attribute is set to `true`, `power`, or `timing`, a "Bit-strength (strong to weak)" column will also be added in the report. *Data_type*: bool, optional

`-latches`	When specified, reports multi-bit latches in the design similar to the reporting of MBFFs. All other parameters work with this paramater just as they work with the flip-flops. When this parameter is specified with the below parameters, the follwowing information about the latches is reported: • `-all`: Reports all the multi-bit latches in the design. • `-cell` *`cell_name`*: Specifies the name of the latch for which information is reported. You can specify both, single-bit and multi-bit latch for reporting this information. • `-equivalence`: Reports multi-bit latch equivalence. It reports all the equivalent multi-bit latch cells and corresponding single-bit latch cells in the design. This parameter is used along with the `-all` parameter. • `-info`: This parameter reports the following properties of all the multi-bit latch cells in the design: *Name, NumBites, Don't Use/ Don't Touch, Dimensions, ScanType, Stateless Leakage, Stateless Dynamic, and LibName* This parameters is used along with the `-all` parameter. • `-statistics`: Reports the single-bit and multi-bit latch count in the current design. • `-usage`: Reports the multi-bit latch usages. This parameter reports the following usage of all the multi-bit latch cells in the design: *Name, NumBits, Count, Total Area, Total Leakage, Total Dynamic, and Min-Max Total Power Range* This parameter is used along with the `-all` parameter.
`-not_merged_inst_list`	When specified, reports the number of unmerged instances. *Data_type*: bool, optional
`-out_file` *`file_name`*	Specifies the name of the output file. *Data_type*: string, optional
`-reason_not_merged {partition cellequivalent contsraint scanconnection location timing power user legalization bus all}`	

	Reports the instances that could not be merged for the specified reason. Specify any of the reasons in the argument or specify `all` to report all instances along with the reason why they could not be merged. `partition`: odd number of flops in one partition `cellequivalent`: no usable equivalent multi-bit flip-flop cell constraint: Constraints by sdc, rule, or user settings `scanconnection`: no scan connection to other flops in the same group `location`: too far away with other flops in the same group `timing`: exceed critical path threshold or MBFF merging that degrades timing `power`: all available multi-bit cells causing power degradation `user`: user set to be excluded in flop merge `legalization`: `bus`: non-bus flops `all`: all the reasons of not merged *Data_type*: enum, optional

`-statistics`	Reports the single-bit and multi-bit flip flop count in the current design. The number is computed as follows:
	
	`----------------------------------------------`
	`Current design flip-flop statistics`
	
	`Single-Bit FF Count : 45573`
	
	`Multi-Bit FF Count : 29213`
	`- 2-Bit FF Count : 9068`
	`- 4-Bit FF Count : 4463`
	`- 6-Bit FF Count : 15682`
	`Total Bit Count : 175653`
	`Total FF Count : 74786`
	`Bits Per Flop : 2.349`
	`----------------------------------------------`
	The number is computed as below:
	`Total Bit Count = 45573 + 9068 * 2 + 4463 * 4 + 15682 * 6 = 175653`
	`Total FF Count = 45573 + 29213 = 74786`
	`Bits Per Flop = 175653 / 74786 = 2.349`
	
	Data_type: bool, optional

`-usage`	Reports the multi-bit flip-flop usage. This parameter is used along with the `-all` parameter. This parameter reports the following usage of all the multi-bit flip-flop cells in the design: `Name, NumBits, Count, Total Area, Total Leakage, Total Dynamic,` and `Min-Max Total Power Range`. **Note**: The last three columns of the report will be generated only when this command is called after running the `report_power` command. **Note**: The unit of the numbers reported in the "Total area" column is in terms of DB-unit-square (`dbu^2`). To convert it to `um^2`, use the following conversion: `set areaUm [expr $areaInDbU / [uu2dbu 1] / [uu2dbu 1]]` *Data_type*: bool, optional

Examples

Use the following command to display information about all the flip-flops in the design.

```
report_multi_bit_ffs -all
```

The software returns the following information:

```
All Multi-Bit FlipFlop Cell/Library Related Information:
-------------------------------------------------------------------------------------------
MBFF Cell                       NumBits  DU/DT  Dimensions  ScanType  Stateless Leakage  Stateless Dynamic LibName
MB2SDFCSNQOPPSBD1BWP20P90ULVT   2        F/F    9900*1152   Parallel  4.696074e-10       1.140480e+07 tcbn20socbwp_ecsm
MB2SDFCNQOPPSAD1BWP20P90ULVT    2        F/F    8280*1152   Parallel  5.108591e-10       9.538560e+06 tcbn20socbwp_ecsm
--------
All Multi-Bit FlipFlop Cells' Usage Reporting:
-------------------------------------------------------------------------------------------
MBFF Cell                       NumBits Count  Total Area  Total Leakage Total Dynamic Min-Max Total Power Range
MB2SDFCSNQOPPSBD1BWP20P90ULVT   2           0          -   -             -             -
MB2SDFCNQOPPSAD1BWP20P90ULVT    2           0          -   -             -             -
-----
All Multi-Bit FlipFlop Cells' Equivalence Reporting:
-------------------------------------------------------------------------
Equivalent Multi-Bit FlipFlop Cells Ordered On Size:
-------------------------------------------------------
MB2SDFQOPPSBD1BWP24P90ELVT  MB2SDFQOPPSBD1BWP20P90ULVT  MB2SDFQOPPSAD1BWP20P90LVT
MB2SDFQOPPSBD1BWP20P90LVT  MB2SDFQOPPSAD1BWP24P90ELVT  MB2SDFQOPPSAD1BWP20P90ULVT
-------------------------------------------------------
Corresponding Single-Bit FlipFlop Cells Ordered On Size:
-------------------------------------------------------
SDFQARD1BWP24P90ELVT  SDFQARD1BWP20P90LVT  SDFQARD1BWP20P90ULVT
SDFQD0BWP24P90ELVT  SDFQD1BWP20P90ULVT  SDFQD0BWP20P90ULVT
SDFQD1BWP24P90ELVT  SDFQD0BWP20P90
-----
```

Use the following command to display information about the multi-bit flip-flop for the specified cell:

```
report_multi_bit_ffs -cell MB2SDFCNQOPPSBD1BWP24P90ELVT
```

```
This Multi-Bit FlipFlop Cell/Library Related Information:
------------------------------------------------------------------------------------------------------------------
MBFF Cell                     NumBits  DU/DT Dimensions ScanType StatelessLeakage StatelessDynamic LibName
MB2SDFCNQOPPSBD1BWP24P90ELVT 2        F/F   8280*1152  Parallel 3.606342e-10      9.538560e+06   tcbn20socbwp_ecsm
This Multi-Bit FlipFlop Cell's Usage Reporting:
--- --------------------------------------------------------------------------------------------------------------
MBFF Cell                      NumBits  Count  Total Area Total Leakage Total Dynamic Min-Max Total Power Range
MB2SDFCNQOPPSBD1BWP24P90ELVT   2       0      -          -             -             -

This Multi-Bit FlipFlop Cell's Equivalence Reporting:
-----------------------------------------------------------------------------
Equivalent Multi-Bit FlipFlop Cells Ordered On Size:
-----------------------------------------------------
MB2SDFCNQOPPSBD1BWP24P90ELVT  MB2SDFCNQOPPSBD1BWP20P90ULVT  MB2SDFCNQOPPSAD1BWP20P90LVT
MB2SDFCNQOPPSBD1BWP20P90LVT MB2SDFCNQOPPSAD1BWP24P90ELVT  MB2SDFCNQOPPSAD1BWP20P90ULVT
-----------------------------------------------------
Corresponding Single-Bit FlipFlop Cells Ordered On Size:
-----------------------------------------------------
SDFCNQARD1BWP24P90ELVT  SDFCNQARD1BWP20P90LVT  SDFCNQARD1BWP20P90ULVT  SDFCNQD0BWP24P90ELVT
SDFCNQD1BWP20P90ULVT  SDFCNQOPPSAD1BWP20P90ULVT  SDFCNQD1BWP24P90ELVT  SDFCNQD1BWP20P90
SDFCNQD0BWP20P90  SDFCNQOPPSBD1BWP20P90ULVT  SDFCNQD1BWP20P90LVT  SDFCNQD0BWP20P90LVT
......
```

Related Information

- `eco_clone_flop`
- `eco_split_complex_flop`
- `write_multi_bit_flop_mapping_file`
- The following opt Category Attributes:
  - `opt_multi_bit_flop_opt`
  - `opt_allow_multi_bit_on_flop_with_sdc`
- The following sections in Optimizing Timing chapter of *Innovus Stylus Common UI User Guide*
  - Using the Multi-Bit Flip-Flop Flow
  - Reporting the MBFFs

report_path_group_options

```
report_path_group_options [-help]
```

Reports the values for each parameter (path_group_name, effort_level, slack_adjustment, and target_slack) for each path group. If no path groups are created, the software issues a warning.

To report the names of the path groups, use the `report_path_groups` command.

Parameter

`-help`	Prints a brief description that includes type and default information for the `report_path_group_options` command. For a detailed description of the command, use the man command: `man report_path_group_options`

Example

Use the following set of commands to create the basic path groups, set the parameters, and report the values of these parameters:

```
create_basic_path_groups

Created reg2reg path group

Effort level <high> specified for reg2reg path_group

Created reg2cgate path group
Effort level <high> specified for reg2cgate path_group

set_path_group_options reg2reg -effort_level low -skewing_slack_constraint 0.3 -
view AV_HL_FUNC_MIN_RC1
set_path_group_options reg2cgate -effort_level high-skewing_slack_constraint 0.5 -
view AV_HL_FUNC_MAX_RC1


report_path_group_options
```

The software generates the following report:

```
-------------------------------------------------------------------------------
-------
|Path Group|Effort|Adjustment(early/late)|Target Slack|weight|SkewingSlackCons|view
```

```
      |
-------------------------------------------------------------------------------------------
-------
|reg2reg    | high | 0.1    /-0.2          | 0           |  2    |     0.3
|AV_HL_FUNC_MIN_RC1 |
|reg2cgate | high | 0.2    /0             | 0           |  1    |     0.5
|AV_HL_FUNC_MAX_RC1 |
```

For detailed descriptions of the above columns, see the table in `set_path_group_options`.

Related Information

- `create_basic_path_groups`
- `report_path_groups`
- `reset_path_group_options`
- `set_path_group_options`
- The following section in Optimizing Timing chapter of *Innovus Stylus Common UI User Guide*
  - Using Additional Post-CTS Timing Optimization Parameters

report_route_type_constraints

```
report_route_type_constraints
[-help]
[-out_file file_name]
[-stack_via_rule]
[-stack_via_rule_only]
[-threshold value]
[-summary  | -detailed ] [-detailed  [-timing ]]
```

Reports the route type constraints that are being added on nets during the Stylus flow. It allows you to monitor the number of constraints, the preferred bottom layer used, the adherence of the different constraint type by the router, and the type of non-default rule (NDR) being used.

The command extracts information about the following groups of route type constraints:

- **INNOVUS**: includes all route types being added by the software and that are using a bottom preferred layer and the default rule.
- **INNOVUS_NDR**: includes all route types being added by the software and that are using an NDR.
  `set_db opt_pre_route_ndr_aware` makes this type of constraint.
- **INNOVUS_CLK**: includes all route types of clock nets that are being added by the software and that are using default rule.
  `create_route_type` without `-non_default_rule` and cts category attribute, `cts_route_type` make this type of constraint.
- **INNOVUS_CLK_NDR**: include all route types of clock nets that are being added by the software and that are using an NDR.
  `create_route_type` with `-non_default_rule` and cts category attribute, `cts_route_type` make this type of constraint.
- **USER**: includes all route types being added by the user. This can be done using the `set_route_attributes` command.
- **USER_NDR**: includes all route types being added by the user and that are using an NDR. This can be done using the `set_route_attributes` command.

When via pillar report is enabled, using the `-stack_via_rule`/`-stack_via_rule_only` parameters, the command checks via pillar assigned on data/clock nets and on the secondary PG nets. For sample reports, see Use Model Scenarios.

Parameters

Parameters

`-help`	Outputs a brief description that includes type and default information for each `report_route_type_constraints` parameter. For a detailed description of the command and all of its parameters, use the man command: `man report_route_type_constraints`.
`-detailed`	Prints detailed information on each net with a route type constraints such as the wire length, the routing adherence, some timing information, and so on. **Note**: The parameters, `-detailed` and `-summary` are mutually exclusive. *Data_type*: bool, optional
`-out_file` *file_name*	Specifies the file name in which the results will be written. If this parameter is not specified, the information is written out in the console and the log file. *Data_type*: string, optional
`-stack_via_rule`	When specified, reports pins with stack via rule constraints. *Data_type*: bool, optional
`-stack_via_rule_only`	When specified, reports only those pins that have stack via rule constraints. *Data_type*: bool, optional
`-summary`	Prints a summary of the layer constraint usage per layer, per NDR, and per constraint type. **Note**: The parameters, `-detailed` and `-summary` are mutually exclusive. *Data_type*: bool, optional

`-threshold value`	Specifies the maximum routing adherence for a net to be included in detailed report. This parameter is specified along with the `-detailed` paramater. The default value is 80, which means only nets with adherence < 80 % will be included in detailed report. **Note**: It is recommended that you do not use the `-threshold` parameter along with the `-stack_via_rule_only` parameter because stack via rule has no “routing adherence ratio” metric. *Data_type*: float, optional
`-timing`	When this parameter is specified, the report will display timing information on the nets such as slack, delay, and incremental delay while generating the detailed information. *Data_type*: bool, optional

Use Model Scenarios

The use model scenarios for reports including via pillar constraints applied on secondary PG pins are provided below.

Reporting stack via rule (including stack via rule used on secondary PG pins) and bottom layer constraint:

```
report_route_type_constraints -stack_via_rule
```

The following information is included in the summary report:

```
**INFO (OILAM) : Summary of current layer assignment usage:
        # num nets with bottom preferred layer constraint                  :  38225
        List NDR used                                                      :  TrunkNDR LeafNDR
        List stack via rule used                                           :
        Num nets with incorrect stack via rule constraint applied on driver  :  0
        Num nets with incorrect stack via rule constraint applied on sinks   :  0
        Num library cells with stack via rule defined                      :  0
        Num library cells with required stack via rule defined             :  0
        Num instances with pillar rule list associated to them             :  0
        List stack via rule used on second PG pins                         :  VPPERF_M2_M4_1_3_1 VPPERF_M2_M4_1_2_1
        Num second PG pins with stack via rule constraint                  :  24785
```

Reporting only stack via rule

```
report_route_type_constraints -stack_via_rule_only
```

The above command prints information about only stack via rule including the summary of via pillar rule applied on secondary PG pins.

```
**INFO (OILAM) : Summary number of stack via rule constraint applied on second PG pins:
        +----------------------+--------------+--------------+----------+--------------+
        | Stack via rule name  | Routing Cost | # stack via rule applied               |
        +                      +              +--------------+----------+--------------+
        |                      |              | Not required | Required                |
        +                      +              +              +----------+--------------+
        |                      |              |              | cell pin | instance pin |
        +----------------------+--------------+--------------+----------+--------------+
        | VPPERF_M2_M4_1_3_1   | 104.0        | NA           | 435      | NA           |
        | VPPERF_M2_M4_1_2_1   | 88.0         | NA           | 24350    | NA           |
        +----------------------+--------------+--------------+----------+--------------+
        | Total                |              | 0            | 24785    | 0            |
        +----------------------+--------------+--------------+----------+--------------+
```

When no via pillar are found on 2nd PG pin, the table will be replaced by following line:

```
**INFO (OILAM) : Summary number of stack via rule constraint applied on second PG pins:
        No stack via rule found
```

When the `-detailed` parameter is specified while reporting via pillar usage, a new table is generated in the report with the list of secondary PG pins with a via pillar constraint. This is shown below.

```
**INFO (OILAM) : List of second PG pins with stack via rule constraint 'VPPERF_M2_M4_1_2_1':
        +-------+----------------------+--------------+-----------------------------------------+
        | ID    | Stack via rule name  | Rule type    | Instance PG pin name                    |
        +-------+----------------------+--------------+-----------------------------------------+
        | 1     | VPPERF_M2_M4_1_2_1   | Cell Required | FE_OCPC96413_FE_OFN15677_TC_13/TVDD    |
        | 2     | VPPERF_M2_M4_1_2_1   | Cell Required | FE_OCPC96403_TC_447/TVDD               |
        | 3     | VPPERF_M2_M4_1_2_1   | Cell Required | FE_OFC93880_TC_1999/TVDD               |
        | 4     | VPPERF_M2_M4_1_2_1   | Cell Required | FE_OFC10210_TC_1612/TVDD               |
        | 5     | VPPERF_M2_M4_1_2_1   | Cell Required | FE_OFC10209_TC_1612/TVDD               |
        | 6     | VPPERF_M2_M4_1_2_1   | Cell Required | FE_OFC10208_TC_1379/TVDD               |
        | 7     | VPPERF_M2_M4_1_2_1   | Cell Required | FE_OFC10207_TC_1379/TVDD               |
        | 8     | VPPERF_M2_M4_1_2_1   | Cell Required | FE_OFC10206_TC_52/TVDD                 |
        | 9     | VPPERF_M2_M4_1_2_1   | Cell Required | FE_OFC10204_TC_776/TVDD                |
        | 10    | VPPERF_M2_M4_1_2_1   | Cell Required | FE_OFC93879_FE_OFN6183_TC_483/TVDD     |
```

Examples

- The following command prints the summary for the route type constraints:

  ```
  report_route_type_constraints -summary
  ```

  The report below summarizes the number of route type constraints per layer for each category.

```
**INFO (OILAM) : Detailed number of layer constraint usage:
+---------+---------+-------------+-------------+-----------------+------+---------+---------+-------------+
| Layer   | INNOVUS | INNOVUS_NDR | INNOVUS_CLK | INNOVUS_CLK_NDR | USER | USER_NDR | UNKNOWN | UNKNOWN_NDR |
+---------+---------+-------------+-------------+-----------------+------+---------+---------+-------------+
| M1(1)   | 0       | 0           | 0           | 0               | 0    | 0       | 0       | 0           |
| M2(2)   | 0       | 0           | 0           | 0               | 0    | 0       | 0       | 0           |
| M3(3)   | 0       | 0           | 1493        | 0               | 0    | 0       | 0       | 0           |
| M4(4)   | 0       | 16          | 0           | 0               | 0    | 0       | 0       | 0           |
| M5(5)   | 0       | 0           | 0           | 1339            | 0    | 0       | 0       | 0           |
| M6(6)   | 0       | 0           | 0           | 0               | 0    | 0       | 0       | 0           |
| M7(7)   | 166     | 20          | 0           | 0               | 0    | 0       | 0       | 0           |
| M8(8)   | 0       | 0           | 0           | 0               | 0    | 0       | 0       | 0           |
| M9(9)   | 25      | 0           | 0           | 0               | 0    | 0       | 0       | 0           |
| M10(10) | 0       | 0           | 0           | 0               | 0    | 0       | 0       | 0           |
| AP(11)  | 0       | 0           | 0           | 0               | 0    | 0       | 0       | 0           |
+---------+---------+-------------+-------------+-----------------+------+---------+---------+-------------+
| Total   | 191     | 36          | 1493        | 1339            | 0    | 0       | 0       | 0           |
+---------+---------+-------------+-------------+-----------------+------+---------+---------+-------------+
```

- The report below summarizes the number of route type constraints per NDR and for each category. The syntax used is *categories_NDR* `name`.

```
**INFO (OILAM) : Detailed number of layer constraint usage per NDR rule:
+---------+----------------------+----------------------+-----------------------------+
| Layer   | INNOVUS_NDR_BUSSPACE | INNOVUS_NDR_WIDEWIRE | INNOVUS_CLK_NDR_DOUBLESPACE |
+---------+----------------------+----------------------+-----------------------------+
| M1(1)   | 0                    | 0                    | 0                           |
| M2(2)   | 0                    | 0                    | 0                           |
| M3(3)   | 0                    | 0                    | 0                           |
| M4(4)   | 14                   | 2                    | 0                           |
| M5(5)   | 0                    | 0                    | 1339                        |
| M6(6)   | 0                    | 0                    | 0                           |
| M7(7)   | 0                    | 20                   | 0                           |
| M8(8)   | 0                    | 0                    | 0                           |
| M9(9)   | 0                    | 0                    | 0                           |
| M10(10) | 0                    | 0                    | 0                           |
| AP(11)  | 0                    | 0                    | 0                           |
+---------+----------------------+----------------------+-----------------------------+
| Total   | 14                   | 22                   | 1339                        |
+---------+----------------------+----------------------+-----------------------------+
```

- The following command prints the detailed route type constraints information:
  `report_route_type_constraints -detailed`

  The report below summarizes the percentage of respect of the route type constraint (bottom preferred layer) by the current router. For example, in the report below, for the INNOVUS constraints between 90% and 100% respect of the constraint, there are 100.0% of the nets. While INNOVUS_CLK constraints between 90 - 100% respect of the constraint, there are 98.33% of the nets.

```
**INFO (OILAM) : Summary min layer respect by router:

+-----------+---------+---------------+-------------+-----------------+-------------------+-------------+-----------------+-------------------+
| Respect   | INNOVUS | INNOVUS M7(7) | INNOVUS_NDR | INNOVUS_NDR M4(4) | INNOVUS_NDR M7(7) | INNOVUS_CLK | INNOVUS_CLK M3(3) | INNOVUS_CLK_NDR |
+-----------+---------+---------------+-------------+-----------------+-------------------+-------------+-----------------+-------------------+
| 0 - 10 %  | 0.00 %  | 0.00 %        | 0.00 %      | 0.00 %          | 0.00 %            | 0.00 %      | 0.00 %          | 5.60 %            |
| 10 - 20 % | 0.00 %  | 0.00 %        | 0.00 %      | 0.00 %          | 0.00 %            | 0.00 %      | 0.00 %          | 1.34 %            |
| 20 - 30 % | 0.00 %  | 0.00 %        | 0.00 %      | 0.00 %          | 0.00 %            | 0.00 %      | 0.00 %          | 1.34 %            |
| 30 - 40 % | 0.00 %  | 0.00 %        | 0.00 %      | 0.00 %          | 0.00 %            | 0.07 %      | 0.07 %          | 1.12 %            |
| 40 - 50 % | 0.00 %  | 0.00 %        | 0.00 %      | 0.00 %          | 0.00 %            | 0.07 %      | 0.07 %          | 0.82 %            |
| 50 - 60 % | 0.00 %  | 0.00 %        | 0.00 %      | 0.00 %          | 0.00 %            | 0.00 %      | 0.00 %          | 1.05 %            |
| 60 - 70 % | 0.00 %  | 0.00 %        | 0.00 %      | 0.00 %          | 0.00 %            | 0.07 %      | 0.07 %          | 1.19 %            |
| 70 - 80 % | 0.00 %  | 0.00 %        | 0.00 %      | 0.00 %          | 0.00 %            | 0.00 %      | 0.00 %          | 3.58 %            |
| 80 - 90 % | 0.00 %  | 0.00 %        | 0.00 %      | 0.00 %          | 0.00 %            | 1.47 %      | 1.47 %          | 8.81 %            |
| 90 - 100 %| 100.00 %| 100.00 %      | 100.00 %    | 100.00 %        | 100.00 %          | 98.33 %     | 98.33 %         | 75.13 %           |
+-----------+---------+---------------+-------------+-----------------+-------------------+-------------+-----------------+-------------------+
```

The report below summarizes the percentage of respect of the NDR constraint by the current router. For example, in the report below, for the INNOVUS_NDR constraints between 90% and 100% respect of the constraint, there are 100% of the nets.

```
**INFO (OILAM) : Summary NDR respect by router:
        +------------+------------+-------------------------+-------------------------+
        | Respect    |INNOVUS_NDR |INNOVUS_NDR_NDR_p10_p10 |INNOVUS_NDR_NDR_p20_p10 |
        +------------+------------+-------------------------+-------------------------+
        | 0 - 10 %   |0.00 %      |0.00 %                   |0.00 %                   |
        | 10 - 20 %  |0.00 %      |0.00 %                   |0.00 %                   |
        | 20 - 30 %  |0.00 %      |0.00 %                   |0.00 %                   |
        | 30 - 40 %  |0.00 %      |0.00 %                   |0.00 %                   |
        | 40 - 50 %  |0.00 %      |0.00 %                   |0.00 %                   |
        | 50 - 60 %  |0.00 %      |0.00 %                   |0.00 %                   |
        | 60 - 70 %  |0.00 %      |0.00 %                   |0.00 %                   |
        | 70 - 80 %  |0.00 %      |0.00 %                   |0.00 %                   |
        | 80 - 90 %  |0.00 %      |0.00 %                   |0.00 %                   |
        | 90 - 100 % |100.00 %    |100.00 %                 |100.00 %                 |
        +------------+------------+-------------------------+-------------------------+
```

The report below provides information similar to above reports but on a net to net basis.

```
 **INFO (OILAM) : List of nets with a constraint from INNOVUS_CLK_NDR and minZ = 5 and routing adherence < 80 %:

+-----+---------------+---------------+-----------------------+-------------+-------------+-------------+-----------------+
| ID  | NDR Cstr      | Min layer Cstr | Route type adherence (%) | Wire Length | X wire length | Y wire length | net             |
+-----+---------------+---------------+-----------------------+-------------+-------------+-------------+-----------------+
| 1   | DOUBLESPACE   | M5(5)         | 0.00                  | 0.81        | 0.00        | 0.81        | CTS_107         |
| 2   | DOUBLESPACE   | M5(5)         | 0.00                  | 1.00        | 0.67        | 0.32        | A/B/CTS_52      |
| 3   | DOUBLESPACE   | M5(5)         | 0.00                  | 1.05        | 0.27        | 0.78        | oreg/stepolkqq  |
| 4   | DOUBLESPACE   | M5(5)         | 0.00                  | 1.32        | 0.95        | 0.37        | A/D/E/CTS_10    |
| 5   | DOUBLESPACE   | M5(5)         | 0.00                  | 2.03        | 1.35        | 0.68        | CTS_20          |
| 6   | DOUBLESPACE   | M5(5)         | 0.00                  | 2.10        | 0.00        | 2.10        | A/CTS_35        |
| 7   | DOUBLESPACE   | M5(5)         | 0.00                  | 2.15        | 1.35        | 0.80        | B/CTS_71        |
...
```

- The following command prints the detailed route type constraints information with specified threshold routing adherence of 101, which means only nets with adherence < 101 % will be included in detailed report.

  ```
  report_route_type_constraints -detailed -threshold 101
  ```

A part of the sample report is shown below.

```
**INFO (OILAM) : List of nets with a constraint from INNOVUS and minZ = 13 and routing adherence < 101 %:
+---+---------------+--------------------------+------------+---------------+---------------+---------------------+
| ID| Min layer Cstr| Route type adherence (%) | Wire Length | X wire length | Y wire length | net                 |
+---+---------------+--------------------------+------------+---------------+---------------+---------------------+
| 1 | M12(13)       | 100.00                   | 480.57     | 480.45        | 0.12          | FE_OFN1_n1          |
| 2 | M12(13)       | 100.00                   | 494.73     | 494.25        | 0.48          | FE_OCPN2_FE_OFN0_n1 |
+---+---------------+--------------------------+------------+---------------+---------------+---------------------+
```

- The following command prints information about the layer assignment, NDR, and stack via rule.

  `report_route_type_constraints -stack_via_rule`

  A part of the sample report is shown below.

```
**INFO (OILAM) : Summary number of stack via rule constraint applied on driver:
+---------------------+--------------+-----------------------------+-----------+-------------+-----------+-----------+-------------+-------------------------+
| Stack via rule name | Routing Cost | # stack via rule applied    |           |             | Adherence |           |             | Bottom layer constraints |
+                     +              +-----------------------------+-----------+-------------+-----------+-----------+-------------+                         +
|                     |              | Not required | Required     |           |             | Not required | Required  |             | M12(13)                 |
+                     +              +              +-----------+-------------+             +-----------+-------------+                         +
|                     |              |              | cell pin  | instance pin |             | cell pin  | instance pin |                         |
+---------------------+--------------+--------------+-----------+-------------+-----------+-----------+-------------+-------------------------+
| VPPERF_M1_M4_2_2_3_1 | 143.0       | 2            | 0         | 0           | 100.00    | NA        | NA          | 2                       |
+---------------------+--------------+--------------+-----------+-------------+-----------+-----------+-------------+-------------------------+
| Total               |              | 2            | 0         | 0           | 100.00    | NA        | NA          | 2                       |
+---------------------+--------------+--------------+-----------+-------------+-----------+-----------+-------------+-------------------------+
```

- The following command prints information about only stack via rule.

  `report_route_type_constraints -stack_via_rule_only`

  A part of the sample report is shown below.

```
**INFO (OILAM) : Summary number of stack via rule constraint applied on sink:
+---------------------+--------------+-----------------------------+-----------+-------------+-----------+-----------+-------------+
| Stack via rule name | Routing Cost | # stack via rule applied    |           |             | Adherence |           |             |
+                     +              +-----------------------------+-----------+-------------+-----------+-----------+-------------+
|                     |              | Not required | Required     |           |             | Not required | Required  |             |
+                     +              +              +-----------+-------------+             +-----------+-------------+
|                     |              |              | cell pin  | instance pin |             | cell pin  | instance pin |
+---------------------+--------------+--------------+-----------+-------------+-----------+-----------+-------------+
| VPPERF_M1_M4_2_2_3_1 | 143.0       | 0            | 0         | 0           | NA        | NA        | NA          |
+---------------------+--------------+--------------+-----------+-------------+-----------+-----------+-------------+
| Total               |              | 0            | 0         | 0           | NA        | NA        | NA          |
+---------------------+--------------+--------------+-----------+-------------+-----------+-----------+-------------+
```

Related Information

- ccopt_design
- opt_design
- opt Category Attributes
- set_route_attributes

report_threshold_instance_count

```
report_threshold_instance_count
[-help]
[-area]
[-detailed]
[-leakage]
[-out_file file_name]
```

Reports the instance distribution across the VT partitions in the design. By default, the Stylus software partitions the design into either two or three VT buckets. Each library, cell, and instance falls into one of these buckets and is reported by this command.

Parameters

`-help`	Prints a brief description that includes type and default information for each `report_threshold_instance_count` parameter. For a detailed description of the command and all of its parameters, use the man command: `man report_threshold_instance_count`
`-area`	Prints the area information across libraries in each VT partition. *Data_type*: bool, optional
`-detailed`	Prints complete details in the specified file. **Note**: Before specifying this parameter, you must specify the file name using the `-out_file` parameter. *Data_type*: bool, optional
`-leakage`	Prints the leakage distribution across the libraries in each VT partition. When you specify this parameter, a complete leakage power analysis is performed on the design, which takes a little long to run. *Data_type*: bool, optional
`-out_file` *file_name*	
	Prints the output in the specified file in the format required for user-defined VT partitioning. **Note**: To print the same report that is output to the log that includes all leakage and instance information, specify the `-detailed` parameter. If you do not specify the `-detailed` parameter, the `-out_file` parameter will only print the VT partitions. *Data_type*: string, optional

Examples

- The following command outputs the Vt partitions to the file `VT_Partitions.rpt`:

  ```
  report_threshold_instance_count -out_file Vt_Partitions.rpt
  ```

  The `Vt_Partitions.rpt` file contains the following information:

```
#####################################################################
#  Generated by:      Cadence Encounter <version>
#  OS:                Linux x86_64(Host ID <machine name>)
#  Generated on:      <date>
#  Design:            <design name>
#  Command:           report_threshold_instance_count -out_file VT_Partitions.rpt
#####################################################################

LVT
at28nm_lvt_aob_s_p125_11v
at28nm_lvt_core_s_p125_11v
at28nm_lvt_nscan_s_p125_11v
at28nm_lvt_scan_s_p125_11v

SVT
at28nm_svt_core_s_p125_11v

HVT
at28nm_svt_aob_s_p125_11v
at28nm_svt_iso_s_p125_11v
at28nm_svt_nscan_s_p125_11v
at28nm_svt_scan_s_p125_11v
at28nm_hvt_aob_s_p125_11v
at28nm_hvt_core_s_p125_11v
at28nm_hvt_nscan_s_p125_11v
at28nm_hvt_scan_s_p125_11v
```

- The following commands output the Vt distribution as well as information on instances and cells. When the output file is not specified, the `report_threshold_instance_count` command outputs only to the log. The second command outputs both to the log and to the output file.

  ```
  report_threshold_instance_count
  report_threshold_instance_count -detailed -out_file VT_detailed.rpt
  ```

  The report will contain the following information:

```
Power view                = MAX1
Number of VT partitions   = 3
Standard cells in design = 1105
Instances in design       = 45515

Instance distribution across the VT partitions:

 LVT : inst = 0 (0.0%), cells = 346 (31%)
   Lib at28nm_lvt_aob_s_p125_11v : inst = 0 (0.0%)
   Lib at28nm_lvt_core_s_p125_11v : inst = 0 (0.0%)
   Lib at28nm_lvt_nscan_s_p125_11v : inst = 0 (0.0%)
   Lib at28nm_lvt_scan_s_p125_11v : inst = 0 (0.0%)

 SVT : inst = 20232 (44.5%), cells = 296 (27%)
   Lib at28nm_svt_core_s_p125_11v : inst = 20232 (44.5%)

 HVT : inst = 23030 (50.6%), cells = 400 (36%)
   Lib at28nm_svt_aob_s_p125_11v : inst = 10 (0.0%)
   Lib at28nm_svt_iso_s_p125_11v : inst = 325 (0.7%)
   Lib at28nm_svt_nscan_s_p125_11v : inst = 854 (1.9%)
   Lib at28nm_svt_scan_s_p125_11v : inst = 18742 (41.2%)
   Lib at28nm_hvt_aob_s_p125_11v : inst = 2 (0.0%)
   Lib at28nm_hvt_core_s_p125_11v : inst = 2397 (5.3%)
   Lib at28nm_hvt_nscan_s_p125_11v : inst = 0 (0.0%)
   Lib at28nm_hvt_scan_s_p125_11v : inst = 700 (1.5%)
```

- The following commands output reports with the leakage power numbers included. The first command outputs only to the log. The second command outputs both to the log and to the output file.

  ```
  report_threshold_instance_count -leakage
  report_threshold_instance_count -leakage -detailed -out_file
  VT_detailed_leakage.rpt
  ```

  The report will contain the following information:

```
Power view                = MAX1
Number of VT partitions   = 3
Standard cells in design  = 1105
Instances in design       = 45515
Design leakage power      = 1.041 mW

Instance and leakage distribution across the VT partitions:

 LVT : inst = 0 (0.0%), leakage = 0.000 mW (0.0%), cells = 346 (31%)
   Lib at28nm_lvt_aob_s_p125_11v : inst = 0 (0.0%), leakage = 0.000 mW (0.0%)
   Lib at28nm_lvt_core_s_p125_11v : inst = 0 (0.0%), leakage = 0.000 mW (0.0%)
   Lib at28nm_lvt_nscan_s_p125_11v : inst = 0 (0.0%), leakage = 0.000 mW (0.0%)
   Lib at28nm_lvt_scan_s_p125_11v : inst = 0 (0.0%), leakage = 0.000 mW (0.0%)

 SVT : inst = 20232 (44.5%), leakage = 0.240 mW (23.1%), cells = 296 (27%)
   Lib at28nm_svt_core_s_p125_11v : inst = 20232 (44.5%), leakage = 0.240 mW (23.1%)

 HVT : inst = 23030 (50.6%), leakage = 0.631 mW (60.6%), cells = 400 (36%)
   Lib at28nm_svt_aob_s_p125_11v : inst = 10 (0.0%), leakage = 0.000 mW (0.0%)
   Lib at28nm_svt_iso_s_p125_11v : inst = 325 (0.7%), leakage = 0.007 mW (0.7%)
   Lib at28nm_svt_nscan_s_p125_11v : inst = 854 (1.9%), leakage = 0.022 mW (2.1%)
   Lib at28nm_svt_scan_s_p125_11v : inst = 18742 (41.2%), leakage = 0.582 mW (55.9%)
   Lib at28nm_hvt_aob_s_p125_11v : inst = 2 (0.0%), leakage = 0.000 mW (0.0%)
   Lib at28nm_hvt_core_s_p125_11v : inst = 2397 (5.3%), leakage = 0.009 mW (0.9%)
   Lib at28nm_hvt_nscan_s_p125_11v : inst = 0 (0.0%), leakage = 0.000 mW (0.0%)
   Lib at28nm_hvt_scan_s_p125_11v : inst = 700 (1.5%), leakage = 0.010 mW (1.0%)
```

Related Information

- opt_design
- time_design
- opt Category Attributes

reset_path_group_options

```
reset_path_group_options
[-help]
[path_group_name]
[-early]
[-effort_level]
[-late]
[-skewing_slack_constraint]
[-slack_adjustment]
[-slack_adjustment_priority]
[-target_slack]
[-view]
```

Resets the parameters of a specified path group, or of all path groups, to their default values.

- If no group name and no parameters are provided, the command resets all parameters for all path groups to their default values.
- If only the group name is provided, the command resets all parameters of that path group to their default values.
- If a path group and a parameter are provided, then only that parameter of that path group is reset to its default value.

See `set_path_group_options` for parameter descriptions.

Note: This command does not provide the option to reset the weights.
Use `set_path_group_options path_group_name -weight 0` to reset the weights.

Parameters

`-help`	Prints a brief description that includes type and default information for each `reset_path_group_options` parameter. For a detailed description of the command and all of its parameters, use the man command: `man reset_path_group_options`
`path_group_name`	Specifies a path group for which to reset parameters. *Data_type*: string, optional
`parameter_names`	Specifies the parameters to reset to the default value.

Examples

- To reset all parameters of all path groups to the default values, specify the following command:
  ```
  reset_path_group_options
  ```
- To reset all the parameters of the `reg2reg` path group to their default values, but leave the parameters of all other path groups unchanged, specify the following command:
  ```
  reset_path_group_options reg2reg
  ```

Related Commands

- `create_basic_path_groups`
- `group_path`
- `report_path_group_options`
- `reset_path_group`
- `set_path_group_options`

route_opt_design

```
route_opt_design

[-help]
[-drv]
[-hold]
[-ideal_clock]
[-num_paths number_of_paths]
[-opt]
[-report_dir directory_name]
[-report_prefix file_name_prefix]
[-route]
[-setup]
[-timing_debug_report]
```

Combines the routing and post-Route optimization flows into a single flow. The benefit of this approach is to interleave the optimization step with routing step. When this command is run, the software performs global and detail routing, setup/hold timing optimization, and DRV fixing. The `route_opt_design` command can be used instead of using two separate commands listed below:

- `route_design`
- `opt_design -post_route -setup -hold`

Note: The `route_opt_design` flow is different from `route_design` followed by `opt_design -post_route -setup -hold` in that the `route_opt_design` command performs more optimization during the routing stage.

The `route_opt_design` command has parameters to split the routing part (`-route`) and the optimization part (`-opt`). When these parameters are specified, the flow exits the interleaved mode, and goes back to the traditional flow with separate commands. This means that `route_opt_design -route` is equivalent to `route_design`, and `route_opt_design -opt` is equivalent to `opt_design -post_route -setup -hold`.

By default, the optimization of `route_opt_design` fixes both setup and hold timing in the interleaved mode with routing. To fix setup timing only, use `route_opt_design -setup`. When the `-setup` parameter is specified, the software performs setup timing optimization, and skips hold timing fixing.

Any partial optimization, using the `-drv` or `-hold` parameters cannot be worked in the interleaved mode. Therefore, if any of these parameters are specified, the software redirects to the `opt_design -post_route` command with the specified option.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `route_opt_design` parameter. For a detailed description of the command and all of its parameters, use the man command: `man route_opt_design`.
`-drv`	Fixes the max_cap and max_tran violations by using the post-Route optimization flow. This is equivalent to `opt_design -post_route -drv`. **Note**: • Ensure that the design is routed before using this parameter. • The `-drv` parameter can only be used with the `-opt` parameter. *Data_type*: bool, optional
`-hold`	When this parameter is specified without `-setup`, the software exits the interleaved mode, and fixes hold timing violations with the post-Route optimization flow. This is equivalent to `opt_design -post_route -hold`. **Note**: Ensure that the design is routed before using this parameter. When this parameter is specified along with the `-setup` parameter, either `-hold -setup` or `-setup -hold`, both the setup and hold timing violations are fixed in the interleaved mode. This is the default behavior of `route_opt_design`. **Note**: • The design should be in the post CTS stage to use `route_opt_design`. • The `-hold` and `-drv` parameters are mutually exclusive. • The `-hold` parameter can only be used with the `-opt` parameter. *Data_type*: bool, optional

`-ideal_clock`	Performs timing optimization using ideal clocks instead of propagated clocks. Using ideal clocks enables you to optimize timing without building a clock tree first, or ignore the existing clock tree, to close ideal timing. If this parameter is not specified, the software assumes the clock tree has been built, and optimizes timing based on propagated clocks. *Data_type*: bool, optional
`-num_paths` *number_of_paths*	Specifies the number of timing paths generated in the timing report. By default, all timing paths are reported. *Data_type*: integer, optional
`-report_dir` *directoryName*	Specifies the directory where the software writes timing reports generated when this command runs. *Default*: `timingReports` *Data_type*: string, optional
`-opt`	When specified, uses the `opt_design -post_route -setup -hold` command and fixes both setup and hold violations for the post-Route optimization flow. The software exits the interleaved mode, and skips routing. **Note**: Ensure that the design is routed before using this parameter. By default, both setup and hold timing violations are fixed. To limit the scope of the optimization, use the `-setup`, `-hold`, and `-drv` parameters along with `-opt` parameter. **Note**: The parameters `-route` and `-opt` are mutually exclusive. *Data_type*: bool, optional
`-report_prefix` *fileNamePrefix*	
	Specifies a prefix for `route_opt_design` report file names. *Default:* *`DesignName_DesignStage`*, where *`DesignStage`* is `post-CTS`, or `post-Route` for `route_opt_design`. *Data_type*: string, optional

`-route`	Specifies that only routing should be done. This is equivalent to `routeDesign` without performing track optimization in the interleaved mode. **Note**: The design should be in the post CTS stage. Note: The -route parameter and the following parameters are mutually exclusive; `-opt`, `-setup`, `-hold`, and `-drv`. *Data_type*: bool, optional
`-setup`	Corrects setup violations in the interleaved mode. When specified without the `-hold` parameter, the software skips the fixing of hold timing violations. **Note**: The design should be in the post CTS stage. **Note**: The `-setup` and `-drv` parameters are mutually exclusive. *Data_type*: bool, optional
`-timing_debug_report`	When specified, generates a machine-readable timing debug report after completing routing and post-route optimization. This report is used for debugging timing results using the timing debug feature in Stylus. **Note**: You can specify the number of paths to be reported using the `-num_paths` parameter. *Data_type*: bool, optional

Examples

- Use the following command to enable interleaved track optimization during the routing stage, for fixing setup violations only:

  ```
  route_opt_design -setup
  ```

- Use the following command to enable interleaved track optimization during the routing stage, for fixing both setup and hold violations:

  ```
  route_opt_design
  ```

  ```
  route_opt_design -setup -hold
  ```

- Use the following command to exit interleaved track optimization mode and perform routing only:

  ```
  route_opt_design -route
  ```

- Use the following command to exit interleaved track optimization mode and perform post-Route optimization only for fixing setup timing:

  ```
  route_opt_design -opt -setup
  ```

  This is equivalent to `opt_design -post_route` or `opt_design -post_route -setup`

- Use the following command to exit interleaved track optimization mode and perform post-Route optimization for fixing both setup and hold timing:

  ```
  route_opt_design -opt
  ```

  This is equivalent to `opt_design -post_route -setup -hold`

Related Information

- `route_design`
- `opt_design`
- The following topic in Optimizing Timing chapter of *Innovus Stylus Common UI User Guide*
  - Using the route_opt_design Flow

set_buffer_footprint

```
set_buffer_footprint
[-help]
names
[-library libName]
```

Sets the buffer cell footprint name. This footprint name is usually assigned during design import. The software uses buffer footprint names during timing optimization and top-level buffer insertion in a partitioned design. You can specify the library that contains the footprint.

Use this command before running timing optimization.

Parameters

`-help`	Prints a brief description that includes type and default information for each `set_buffer_footprint` parameter. For a detailed description of the command and all of its parameters, use the man command: `man set_buffer_footprint`
names	Specifies the buffer cell footprint name or a buffer cell name. When you specify a buffer cell name the Innovus software automatically finds the cell footprint name. *Data_type*: string, required
`-library` *lib_name*	Specifies the library that contains the buffer footprint. You must insert a colon (`:`) between the library name and the buffer name when you use this parameter. *Data_type*: string, optional

Example

- The following command sets the buffer cell footprint name to `B_1P`:

  ```
  set_buffer_footprint B_1P
  ```

Related Information

- `check_footprint`
- `read_footprint`
- `report_footprints`
- `set_delay_footprint`
- `set_inverter_footprint`
- The following section in Optimizing Timing chapter of *Innovus Stylus Common UI User Guide*
  - Using Cell Footprints

set_delay_footprint

```
set_delay_footprint
[-help]
[-library lib_name]
names
```

Specifies the delay cell footprint. This footprint name is usually assigned during design import. The delay cells from the cell footprint name are used for inserting delays when you run timing optimization to fix hold-time violations.

Parameters

`-help`	Prints a brief description that includes type and default information for each `set_delay_footprint` parameter. For a detailed description of the command and all of its parameters, use the man command: `man set_delay_footprint`
`-library` *lib_name*	Specifies the library name from where the cells will be picked up. *Data_type*: string, optional
names	Specifies the delay cell footprint name or a delay cell name. When you specify a delay cell name, the software automatically finds the cell footprint name. *Data_type*: string, required

Example

The following command sets the delay footprint name to `DEL_1P`:

```
set_delay_footprint DEL_1P
```

Related Information

- `check_footprint`
- `read_footprint`
- `report_footprints`
- `set_buffer_footprint`
- `set_inverter_footprint`
- The following sections in Optimizing Timing chapter of *Innovus Stylus Common UI User Guide*
    - Using Cell Footprints

set_inverter_footprint

```
set_inverter_footprint
[-help]
[-library lib_name]
names
```

Specifies the inverter cell footprints. The footprint names are usually assigned during design import. The Innovus software uses these inverter footprint names during timing optimization and top-level inverter insertion in a partition design.

Parameters

`-help`	Prints a brief description that includes type and default information for each `set_inverter_footprint` parameter. For a detailed description of the command and all of its parameters, use the man command: `man set_inverter_footprint`
names	Specifies the inverter cell footprint names or inverter cell names. When you specify an inverter cell name, the Innovus software automatically finds the cell footprint name. *Data_type*: string, required
`-library` *lib_name*	Specifies the library name from where the cells will be picked up. *Data_type*: string, optional

Related Information

- `check_footprint`
- `read_footprint`
- `report_footprints`
- `set_buffer_footprint`
- `set_delay_footprint`
- The following sections in Optimizing Timing chapter of *Innovus Stylus Common UI User*

Guide

- Using Cell Footprints

set_max_cap_per_freq

```
set_max_cap_per_freq
[-help]
-cap {cap1 cap2 …}
[-force]
-freq {freq1 freq2 …}
[-lib lib_name]
-pins {cell pin}
```

Adds a maximum capacitance per frequency table to the existing `max_capacitance` statement in the timing library. Timing analysis uses this table to detect violations, and `opt_design` attempts to repair violations.

The timing engine derives the frequency of an instance from the clock domain to which it is connected (the fastest clock domain, if there is more than one). If the instance is a clock instance (that is, part of the clock tree), the actual clock frequency is used. If the instance is part of the combinational logic (that is, a flip-flop, or the logic connected to the output of a flip-flop), the frequency used equals one-half of the associated clock domain frequency because a flip-flop output can only toggle at one-half the rate of the input clock. For example, a clock of 1.0 Mhz has 1,000,000 rise and fall transitions per second; however the output of a flip-flop connected to a 1.0 Mhz clock can only have 500,000 rise and fall transitions per second.

Timing analysis computes the frequency directly from the timing graph. Use the `check_ac_limits -toggle` command to write the effective frequency onto the associated net in the database.

Note: This command supports MMMC and performs DRV computation as per view frequency. This feature is enabled by setting the below attribute to `true`.

```
set_global timing_report_drv_enable_frequency_per_view true
```

For detailed description of the variable, see the timing Category Attributes chapter.

You can use this command any time after loading the timing library files (that is, after importing or restoring the design).

Parameters

`-help`	Outputs a brief description that includes type and default information for each `set_max_cap_per_freq` parameter. For a detailed description of the command and all of its parameters, use `man set_max_cap_per_freq`.

`-cap {`*`cap1 cap2 …`*`}`	
	Specifies a list of capacitance values that must be greater than or equal to 0. You must specify the same number of -freq and -cap values. The software derives the maximum capacitance from a linear interpolation (or extrapolation) for a given frequency value using two frequency values in the table. For interpolation, the software uses the closest value above the frequency, and the closest value below the frequency. For extrapolation, the software uses the two closest values to the frequency. *Type:* Float, specified in units of pF *Data_type*: list, required
`-force`	Computes the maximum capacitance from the interpolated `-freq` and `-cap` values only, and ignores any previously defined `max_capacitance` values for these pins in the library. *Default:* Computes the maximum capacitance using the worst case (minimum value) between the library `max_capacitance` values and the interpolated `-cap` values. *Data_type*: bool, optional
`-freq {`*`freq1 freq2 …`*`}`	
	Specifies a list of at least two frequency values that must be greater than or equal to 0. You must specify the same number of -freq and -cap values, and the frequency values must be specified in ascending order. *Type:* Float, specified in units of Hz *Data_type*: list, required
`-lib` *`libName`*	
	Modifies only the matching cell-pins in the specified library. *Default:* Modifies the matching cell-pins in all libraries *Data_type*: string, optional
`-pins {`*`cell pin`*`}`	

	Adds the new capacitance constraints to the specified set of cells and pins. You can specify output or inout pins. The asterisk (*) wildcard character can be used in a cell or pin name. If a table associated with the pin already exists, it is replaced with the new table. For analysis and optimization, the software uses the most restrictive (the lowest capacitance) value from either the table or the .lib max_capacitance. If you also specify the -force parameter, the max_capacitance value is ignored. The final value can be further modified using .sdc set_max_capacitance statements. *Data_type*: list, required

Examples

Assume all output pins of a library have a `max_capacitance` of 3.5 pF from the library file.

- The following command sets the frequency dependent `max_capacitance` to 3.0 pF at a frequency of 100 Mhz, and to 1.5 pF at a frequency of 200 Mhz, for all output or inout pins for all cells in the library named `fast`:

  ```
  set_max_cap_per_freq -lib fast -freq {100e6 200e6} -cap {3.0 1.5} -pins {* *}
  ```

  As a result, for a net at 300 Mhz, an and2 cell output pin has an extrapolated `max_capacitance` value of 0 pF (which means the cell cannot be used at that frequency). An extrapolation to 50 Mhz results in a `max_capacitance` value of 3.75 pF. However, the 3.75 pF value is greater than the existing `.lib max_capacitance` limit of 3.5 pF; therefore timing analysis uses the 3.5 pF value.

- The following command applies the same constraints to all libraries with cell names matching `dff*` that have a pin named `Qbar`:

  ```
  set_max_cap_per_freq -force -freq {100e6 200e6} -cap {3.0 1.5} -pins {dff* Qbar}
  ```

  Note: The original `.libmax_capacitance` limit is ignored because the `-force` parameter is specified; therefore extrapolation to 50 Mhz results in a final `max_capacitance` constraint value of 3.75 pF.

Related Information

- `check_ac_limits`
- `opt_design`
- opt Category Attributes
- timing Category Attributes

set_max_cap_per_freq_transition

```
set_max_cap_per_freq_transition
[-help]
-cap {{cap11 cap12 cap13…} {{cap21 cap22 cap23…}} {{cap31 cap32 cap33…}}}
[-force]
-freq {freq1 freq2 freq3…}
[-input_pin inputpin]
[-lib lib_name]
-pins {cell outputpin}
-transition_time {tran1 tran2 tran3…}
```

Sets the maximum capacitance of a pin in the design. Considers both the input transition time and the output frequency. If the maximum capacitance exceeds this value, the `report_constraint -drv_violation_type max_capacitance` command displays a capacitance violation.

You can use this command any time after loading the timing library files (that is, after importing or restoring the design).

Parameters

`-help`	Outputs a brief description that includes type and default information for each `set_max_cap_per_freq_transition` parameter. For a detailed description of the command and all of its parameters, use `man set_max_cap_per_freq_transition`.
`-cap {{cap11 cap12 cap13…} {{cap21 cap22 cap23…}} {{cap31 cap32 cap33…}}}`	

	Specifies a table of capacitance values with the maximum output pin capacitances associated with `-freq` and `-transition_time` values. The total number of `-cap` values must be the total number `-freq` values times the total number of `-transition_time` values. The `-freq` and `-transition_time` values must be specified before `-cap`. The capacitance values are floats in units of pico farad and all values must be greater than or equal to zero. The capacitance table uses frequency as the first index and the input transition as the second index. For example, `cap11`, `cap12`, and `cap13` are for `freq1`, while `c12`, `c22`, and `c32` correspond to the input transition `tran2`. The maximum capacitance is derived from a linear interpolation (or extrapolation) for a given frequency value and a transition value using two frequency values and two transition values n the table (closest above and below for interpolation, the two closest for extrapolation. *Type:* Float, specified in units of pF *Data_type*: list, required
`-force`	Computes the maximum capacitance from the interpolated `-freq, -transition_time,` and `-cap` values only, and ignores any previously defined `max_capacitance` values for these pins in the library. *Default:* (`-force` not specified) Computes the maximum capacitance using the worst case (minimum value) between the library `max_capacitance` values and the interpolated `-cap` values. *Data_type*: bool, optional
`-freq {`*`freq1 freq2 freq3…`*`}`	
	Specifies a list of at least two frequency values with the toggle frequencies at the output pin. Frequency values must be increasing, and the total number of frequency values cannot exceed 255. *Type:* Float, specified in units of Hz. All values must be equal to or greater than `0`. *Data_type*: list, required
`-input_pin` *`inputpin`*	

	Specifies the input pin name corresponding to the input transition values in `-transition_time`. If *`-inputpin`* is not given, `-transition_time` values correspond to all input pins. *Data_type*: string, optional
`-lib` *`lib_name`*	Specifies the library name from inside a `.lib` file to update the cell-pins. *Default:* If a library name is not specified, modifies the matching cell-pins in all libraries *Data_type*: string, optional
`-pins {`*`cell`* `output`*`pin`*`}`	
	Adds the new capacitance constraints to the specified set of cells and pins. You can specify output or inout pins; input pins are ignored. The asterisk (`*`) wildcard character can be used in a cell or pin name. If there is already a table associated with the same input-output pair (input pins are specified using `-input_pin`), it is replaced with the new constraints; otherwise, it is added to the output maximum capacitance table list. For analysis and optimization, the software uses the most restrictive (the lowest capacitance) value from either the table or the `.lib max_capacitance`. If you also specify the `-force` parameter, the `max_capacitance` value is ignored. The final value can be further modified using `.sdc set_max_capacitance` statements. *Data_type*: list, required
`-transition_time {`*`tran1 tran2 tran3…`*`}`	
	Specifies a list of at least two transition values with the input transition times. Transition times must be increasing in value, and the total number of transition times cannot exceed 255. *Type:* Float, specified in units of nanoseconds, and all values must be equal to or greater than `0`. *Data_type*: list, required

Example

The following command establishes a maximum capacitance table for all pins on all cells in all libraries.

```
set_max_cap_per_freq_transition -force -pins {* *} -freq {1e8 1.25e8 1.5e8} -
transition_time {450 475 500} -cap {{0.022 0.017 0.012} {0.02 0.014 0.01} {0.018 0.013
0.008}}
```

The table is shown below:

	450	475	500
1e8	0.022	0.017	0.012
1.25e8	0.02	0.014	0.01
1.5e8	0.018	0.013	0.008

Related Information

- check_ac_limits
- opt_design
- report_constraint
- opt Category Attributes
- timing Category Attributes

set_max_transition_per_freq

```
set_max_transition_per_freq
[-help]
[-force]
-freq { freq1  freq2 …}
[-lib lib_name]
-pins {cell pin}
-transition {tran1 tran2 ...}
```

Adds a maximum transition per frequency table to the existing `max_transition` statement in the timing library. Timing analysis uses this table to detect violations, and `opt_design` attempts to repair violations.

Use this command any time after loading the timing library files (that is, after importing or restoring the design).

The timing engine derives the frequency of an instance from the clock domain to which it is connected (the fastest clock domain, if there is more than one). If the instance is a clock instance (that is, part of the clock tree), the actual clock frequency is used. If the instance is part of the combinational logic (that is, a flip-flop, or the logic connected to the output of a flip-flop), the frequency used equals one-half of the associated clock domain frequency because a flip-flop output can only toggle at one-half the rate of the input clock. For example, a clock of 1.0 Mhz has 1,000,000 rise and fall transitions per second; however the output of a flip-flop connected to a 1.0 Mhz clock can only have 500,000 rise and fall transitions per second.

Timing analysis computes the frequency directly from the timing graph. Use the `check_ac_limits -toggle` command to write the effective frequency onto the associated net in the database.

Note: This command supports MMMC and performs DRV computation as per view frequency. This feature is enabled by setting the below attribute to `true`.

```
set_global timing_report_drv_enable_frequency_per_view true
```

For detailed description of the variable, see the timing Category Attributes chapter.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `set_max_transition_per_freq` parameter. For a detailed description of the command and all of its parameters, use `man set_max_transition_per_freq`.
`-force`	Computes the maximum transition from the interpolated `-freq` and `-transition` values only and ignores any previously defined `max_transition` values for these pins in the library. *Default:* Uses the worst case (minimum value) between the `.lib max_transition` values and the interpolated `-transition` values. *Data_type*: bool, optional
`-freq {`*`freq1 freq2 …`*`}`	
	Specifies a list of at least two frequency values that must be greater than or equal to 0. You must specify the same number of `-freq` and `-transition` values, and the frequency values must be specified in ascending order. *Type:* Float, specified in units of Hz *Data_type*: list, required
`-lib` *`lib_name`*	
	Modifies only the matching cell-pins in the specified library. *Default:* Modifies the matching cell-pins in all libraries. *Data_type*: string, optional
`-pins {`*`cell pin`*`}`	

	Adds the new transition constraints to the specified set of cells and pins. You can specify input, output, or inout pins. The asterisk (`*`) wildcard character can be used in a cell or pin name. If a table associated with the pin already exists, it is replaced with the new table. For analysis and optimization, the software uses the most restrictive (the lowest transition) value from either the table or the `.lib max_transition`. If you also specify the -force parameter, the `max_transition` value is ignored. The final value can be further modified by using the `.sdc set_max_transition` statements. *Data_type*: list, required
`-transition {tran1 tran2 …}`	Specifies a list of transition values that must be greater than or equal to 0. You must specify the same number of `-freq` and `-transition` values. The software derives the maximum transition limit from a linear interpolation (or extrapolation) for a given frequency value using two frequency values in the table. For interpolation, the software uses the closest value above the frequency, and the closest value below the frequency. For extrapolation, the software uses the two closest values to the frequency. *Data_type*: list, required

Examples

Assume all output pins of a library have a `max_transition` value of 0.9 ns from the library file.

- The following command sets the frequency dependent `max_transition` to 1.0 ns at a frequency of 100 Mhz, and to 0.7 ns at a frequency of 200 Mhz, for pins matching `in*` of cells matching `*` (that is, all cells) in the library named `slow`:

  ```
  set_max_transition_per_freq -lib slow -freq {100e6 200e6} -transition {1.0 0.7} -pins {* in*}
  ```

  As a result, at 300 Mhz, an `and2` cell input pin named `in1` has an extrapolated `max_transition` value of 0.4ns. An extrapolation to 50 Mhz results in a `max_transition` value of 1.15 ns. However, the 1.15 ns value is greater than the existing `.lib max_transition` value of 0.9 ns; therefore timing analysis uses the 0.9 value.
- The following command applies the same constraints to all libraries with cell names matching `dff*` that have a pin named `clk`:

```
set_max_transition_per_freq -force -freq {100e6 200e6} -transition {1.0 0.7} -pins
{dff* clk}
```

Note: The original `.libmax_transition` value is ignored because the `-force` parameter is specified; therefore an extrapolation to 50 Mhz results in a final `max_transition` limit of 1.15 ns.

Related Information

- `set_max_cap_per_freq`
- `check_ac_limits`
- `opt_design`
- `report_constraint`
- opt Category Attributes
- timing Category Attributes

set_path_group_options

```
set_path_group_options

[-help]
path_group_name
[-early]
[-effort_level {low | high}]
[-late]
[-skewing_slack_constraint value_in_ns]
[-slack_adjustment float]
[-slack_adjustment_priority integer]
[-target_slack float]
[-view view_names]
[-weight integer]
```

Specifies options to guide timing optimization and timing analysis when path groups are used. To reset parameters to their default value, use `reset_path_group_options`.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `set_path_group_options` parameter. For a detailed description of the command and all of its parameters, use the man command: `man set_path_group_options`
`-early`	Specifies early mode. This parameter is used with the `-slack_adjustment` option when you wish to adjust a path groups slack in Hold timing mode only. **Note the following**: • If you do not specify either `-early` or `-late`, then `-late` is implied. • If you specify `-early`, then `-early` is used. • If you specify `-late`, then `-late` is used.

- If you specify both `-early` and `-late`, then both `-early` and `-late` are used, which means the same value of `-early` and `-late` are set at the same time.

For example,

```
set_path_group_options reg2reg -
slack_adjustment -0.3 -early -late

Slack adjustment of -0.3 applied on
reg2reg path_group

**INFO: Setting reg2reg as high effort
level since it has a negative slack
adjustment.

[DEV]innovus 25>
report_path_group_options


-----------------------------------------
-----------------------------------------
-----

|Path
Group|Effort|Adjustment(early/late)|Target
Slack|weight|SkewingSlackCons|view|

-----------------------------------------
-----------------------------------------
----

|reg2reg   | high |    -0.3 /     -0.3
|        0   |  2   |        300
|setup_view|
```

Data_type: bool, optional

`-effort_level {low | high}`	Specifies the effort applied to close timing on a specific path group. This parameter is honored by timing optimization (`opt_design`). Select one of the following options: `low`: Specifies that this path group is optimized together with other path groups for best overall slack. low is the default effort value for all user-created path groups. **Note**: Cadence recommends you specify low effort for all path groups except that ones that the tool should focus on most strongly. `high`: Specifies that this path group is optimized separately from the low-effort path groups in order to achieve the best possible WNS for this path group. *Default*: low *Data_type*: enum, optional
path_group_name	Specifies the path group for which to set options. *Data_type*: string, required
`-late`	Specifies late mode. This parameter is used with the `-slack_adjustment` option when you wish to adjust a path groups slack in Setup timing mode only. For usage details, see the description of the `-early` parameter. *Data_type*: bool, optional

`-skewing_slack_constraint` *value_in_ns*	When set, applies the specified skewing slack constraint to the path group. The value specified is in nanoseconds. If not set, no skewing slack constraint is applied to the path group. This parameter provides the ability to control the I/O slack during skewing without degrading the core slack value. The skewing engine skews in a way not to make any already -ve slack path more -ve or not introduce a new -ve slack path. **Examples**: If the following is specified: `set_path_group_options R2O -skewing_slack_constraint 0` *Before skewing*: FF's D slack is `-100 ps`(R2R), Q slack is `-20ps` (R2O) *After skewing*: D slack is `-100ps`, Q slack is `-20ps` *Before skewing*: FF's D slack is `-100 ps`(R2R), Q slack is `30ps` (R2O) *After skewing*: D slack is `-70ps`, Q slack is `0ps` **Note**: • This option can control CCOpt useful skew. If you are running ECF (early clock flow), pre-CTS will use CCOpt useful skew engine to do skewing. in this scenario, this option will take effect. • This option can control post-CTS `opt_clock_skew` behavior. • This option does not impact timing analysis. *Data_type*: float, optional

`-slack_adjustment` *float*	Specifies a positive or negative value used to modify path constraints on all paths that are part of a specific path group. The value is in nanoseconds. To relax the path constraints computed by the timing engine, specify a positive value. To tighten the timing constraints, specify a negative value. This parameter is honored by timing optimization (`opt_design`) and timing analysis (`report_timing`). This means that that the `-slack_adjustment` parameter setting impacts the timing analysis result and all optimization is then performed based on this timing. Using this parameter might have an impact on run time during timing optimization. **Note**: Path groups with negative slack adjustment will be automatically set to high effort, unless the `-effort_level` option is explicitly set to "'`low`". For positive slack adjustment, the effort level will remain the same. This means, if the effort is low before setting a positive slack adjustment, the new effort will be low; if the effort is high before setting a positive slack adjustment, the new effort will be high. **Tip**: Use this parameter together with the `-effort` parameter to direct timing optimization focus. *Data_type*: float, optional

`-slack_adjustment_priority` *integer*	Provides priority to a path group for the slack adjustment feature. The paths that are a part of overlapping path groups get slack adjustment from the path group that has a higher priority value. If the number is higher, the priority is higher. In case of multiple overlapping path groups with different slack adjustment values, the most constraining slack adjustment is applied. *Default*: 0 **Note**: The `-slack_adjustment_priority` parameter is ignored if `-slack_adjustment` is not specified. *Data_type*: unit, optional

`-target_slack` *float*	Specifies a target value for setup slack, in nanoseconds, for all paths that are part of a specific path group. The value can be positive or negative. This parameter is honored by timing optimization (`opt_design`). Using this parameter might have an impact on run time during timing optimization. **Note**: Setting this parameter does not impact the timing analysis results, and only setup optimization considers the target slack as the goal of optimization. The path group target slack (`set_path_group_options -target_slack`) and the global setup target slack (`opt_setup_target_slack`) are cumulative. When both are set, global setup target slack adds on top of what is specified using the `set_path_group_options -target_slack` parameter. The effective number depends upon whether path group target slack number is positive or negative. The global setup target slack is the timing closure target for timing optimization. **Note**: When both, `-slack_adjustment` and `-target_slack` parameters are set, the setup optimization considers the target slack based on slack adjustment timing. *Data_type*: float, optional
`-view` *view_names*	Specifies the view names. This parameter is used when you want to specify different effort levels of the path group for different views. *Default*: `{}` *Data_type*: string, optional

`-weight integer`	Defines the optimization priority. For the default path groups, the software assigns weights in the following manner: For high-effort path groups - the software assigns a weight of "1" to reg2cgate and "2" to reg2reg. All low-effort path groups are assigned a default weight of "0". You can assign any value to this parameter (`0` to `INT_MAX`). There is no upper limit. The software looks at the relative priority of the weights for each path group - the higher the number, the higher the weight. For example, if there are 10 path groups with weights ranging from 1 to 10 and they are all high-effort path groups, the high-effort optimization for all these ten path groups will be started simultaneously. If, however, the optimization is stuck at the path group with the weight 6, the software drops all the path groups that have weights ranging from 1 to 6 and continues to work on the remaining four path groups. **Note**: Use this parameter to control optimization and not `-slack_adjustment_priority`, which only controls the slack adjustment. *Data_type*: unit, optional

Examples

- To add 2 nanoseconds of slack adjustment to all paths belonging to the `PATH_GROUP_A` path group, specify the following command:
  ```
  set_path_group_options PATH_GROUP_A -slack_adjustment 2
  ```
- To subtract 200 picoseconds from all paths belonging to `PATH_GROUP_B` path group, specify the following command:

```
set_path_group_options PATH_GROUP_B -slack_adjustment -0.2
```

- To ensure that the `opt_design` command does not try to achieve a better slack than 2 nanoseconds on path group `PATH_GROUP_A`, specify the following commands:

  ```
  set_path_group_options PATH_GROUP_A -target_slack 2
  opt_design
  ```

  When these commands run, the software internally subtracts 2 nanoseconds of slack adjustment from all paths that belong to `PATH_GROUP_A` path group.

- The following command adds to the slack on all paths inside group `GROUP_1` by 0.2 ns - relaxing the constraints on the paths for Hold timing analysis only:

  ```
  set_path_group_options GROUP_1 -slack_adjustment +0.2 -early
  ```

- The following command subtracts the slack on all paths inside group `GROUP_2` by 0.2 ns - tightening the constraints on the paths for Setup timing analysis only:

  ```
  set_path_group_options GROUP_2 -slack_adjustment -0.2 -late
  ```

- The following commands are used to set the effort level, skewing slack constraint, weight, and views of the two path groups, and generate a report using these parameters:

  ```
  set_path_group_options reg2reg -effort_level low -skewing_slack_constraint 0.3 -
  view AV_HL_FUNC_MIN_RC1
  set_path_group_options reg2cgate -effort_level high-skewing_slack_constraint 0.5 -
  view AV_HL_FUNC_MAX_RC1
  report_path_group_options
  The software prints the following report:
  ------------------------------------------------------------------------------
  -----------------
  |Path Group|Effort|Adjustment(early/late)|Target
  Slack|weight|SkewingSlackCons|view               |
  ------------------------------------------------------------------------------
  -----------------
  |reg2reg   |low   | 0.1     /-0.2        | 0          |   2  |    0.3
   |AV_HL_FUNC_MIN_RC1 |
  |reg2cgate |high  | 0.2    /0            | 0          |   1  |    0.5
   |AV_HL_FUNC_MAX_RC1 |
  ```

  The description of the columns headings and the corresponding parameters to which they map is provided below.

Column Heading	Parameter Name and Description
Path Group	*path_group_name:* Name of the path group for which the options are being set.
Effort	`-effort_level`: Specifies the effort applied to close timing on a specific path group. `low`: Specifies that this path group is optimized together with other path groups for best overall slack. low is the default effort value for all user-created path groups. `high`: Specifies that this path group is optimized separately from the low-effort path groups in order to achieve the best possible WNS for this path group.
Adjustment (early/late)	`-slack_adjustment`: Specifies a positive or negative value used to modify path constraints on all paths that are part of a specific path group. The value is in nanoseconds.
Target Slack	`-target_slack`: Specifies a target value for setup slack, in nanoseconds, for all paths that are part of a specific path group. The value can be positive or negative. **Note**: This column only displays the latest setting, either early or late.
weight	`-weight`: Defines the optimization priority. For the default path groups, the software assigns weights in the following manner: • For high-effort path groups: the software assigns a weight of "1" to reg2cgate and "2" to reg2reg. • All low-effort path groups are assigned a default weight of "0".
Skewing Slack Cons	`-skewing_slack_constraint`: Specifies the skewing slack constraint for the path group. If no constraint is applied to a path group, the column is empty.
view	`-view`: Specifies the view name. It allows you to specify different effort levels of the path group for different views.

Related Commands

- create_basic_path_groups
- group_path
- reset_path_group_options
- report_path_group_options
- reset_path_group

set_scheduling_file

```
set_scheduling_file
[-help]
scheduling_file_name
```

Specifies the name of the scheduling file. The file is not taken into account by CTS, but skew clock must have access to the file if it runs several times consecutively.

To see the current setting for the `set_scheduling_file` command, use `get_scheduling_file`.

Parameters

-help	Outputs a brief description that includes type and default information for each set_scheduling_file parameter. For a detailed description of the command and all of its parameters, use the man command: man set_scheduling_file.
scheduling_file_name	Specifies the name of the scheduling file. *Data_type*: string, required

Example

The following command sets the name of the scheduling file to `scheduling_file.cts`:

```
set_scheduling_file scheduling_file.cts
```

For more information about how the scheduling file is used in the pre-CTS flow, see `opt_clock_skew`.

Related Commands

- `opt_clock_skew`
- `set_scheduling_file`

write_multi_bit_flop_mapping_file

```
write_multi_bit_flop_mapping_file
[-help]
[-map_output_pins {false | true | all}]
[-output directory]
[-prefix fileName_prefix]
```

Dumps the multi-bit flip-flop (MBFF) mapping files in the specified directory with the specified prefix name. The following files are dumped:

- `multi_bit_pin_mapping`
  includes multi-bit input pin mapping information by default. In addition, it appends the output pins when the `-map_output_pins` option is specified.
- `multi_bit_mapping`
  includes multi-bit optimization mapping information

The `multi_bit_pin_mapping` file has two columns. The first column provides the full path name of the flop input D pin as it was in the initial netlist. The second column provides the full path name of flop input D pin in the current state of the database that corresponds to the first column. For example, if SBF1, SBF2, SBF3, SBF4 are single-bit flip flops, and MBF1, and MBF2 are two-bit flip flops, then,

The **multi_bit_mapping** file contains the following:

```
SBF1(SBCell1), SBF2(SBCell2) merged to MBF1(MBCell1)

MBF2(MBCell2) split to SBF3(SBCell3), SBF4(SBCell4)
```

The **multi_bit_pin_mapping** file by default contains the following**:**

```
SBF1/D    MBF1/D1

SBF2/D    MBF1/D2

MBF2/D1   SBF3/D

MBF2/D2   SBF4/D
```

Note:

- The `multi_bit_pin_mapping` file contains all information of the flops whose input D pin names have changed. For example, see the "Examples" section below.
- To generate a proper report for merge/split mapping information, the command should be executed as part of the flow in the live session, for example, after running `place_opt_design`. If

the command is run on a restored session, it will generate an empty "`multi_bit_mapping`" file because the merge/split mapping information is not saved while saving the design.

- You can use the `write_do_lec` command to facilitate the Logical Equivalence Checking (LEC) checking of the MBFF instances. This command writes out a `dofile` that contains the name mapping information of the MBIT flops against their split/single bit candidate.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `write_multi_bit_flop_mapping_file` parameter. For a detailed description of the command and all of its parameters, use the man command: `man write_multi_bit_flop_mapping_file`.
`-map_output_pins {false | true | all}`	When set to `true`, the software writes out only the mapped output terms in the MBFF pin mapping file. When set to `false`, the software writes out only input pin mapping in the pin mapping file. This is the default setting. When set to `all`, the software writes out both input and output pin mapping in the pin mapping file. Example of **multi_bit_pin_mapping** file when `-map_output_pins` parameter is set to `false`: MBF1/D1 SBF1/D MBF1/D2 SBF2/D Example of **multi_bit_pin_mapping** file when the `-map_output_pins` parameter is set to `true`: MBF1/D1 SBF1/D MBF1/D2 SBF2/D MBF1/Q1 SBF1/Q MBF1/Q2 SBF2/Q *Default*: `false`
`-output` *`directory`*	Specifies the directory in which to dump the MBFF mapping files. *Default*: ./ *Data_type*: string, optional
`-prefix` *`fileName_prefix`*	Specifies the prefix to the MBFF mapping files. *Default*: without prefix *Data_type*: string, optional

Examples

Consider a scenario in which the original netlist contains the following information:

```
4BitMBF1          (4 bit MBF, D1-4)

SBF1              (single bit FF)
```

Example of merging is as follows:

If the incoming netlist has `4BitMBF1/D4` dangling (no signal attached) and MERGE, merged `SBF1/D` into `4BitMBF1/D4`. The following line will be included in `multi_bit_input_pin_mapping` file:

```
SBF1/D          4BitMBF1/D4
```

Example of splitting is as follows:

If incoming netlist has valid signals for all 4 bits of `4BitMBF1` and splits `4BitMBF1/D4` out to `SDF2/D` leaving `4BitMBF1/D4` dangling, then the following lines will be included in `multi_bit_pin_mapping` file:

```
*VOID*          4BitMBF1/D4

4BitMBF1/D4     SBF2/D
```

Related Information

- `eco_clone_flop`
- `eco_split_complex_flop`
- `report_multi_bit_ffs`
- `write_do_lec`
- `rename_obj`
- The following opt Category Attributes:
  - `opt_multi_bit_flop_opt`
  - `opt_allow_multi_bit_on_flop_with_sdc`
- The following sections in Optimizing Timing chapter of *Innovus Stylus Common UI User Guide*
  - Using the Multi-Bit Flip-Flop Flow

- Dumping the MBFF File
- Reporting the MBFFs

write_target_based_opt_file

```
write_target_based_opt_file
[-help]
-bottleneck_instance_number integer
[-early]
[-gtd_file filename]
[-max_paths integer]
[-max_slack float]
[-min_slack float]
-out_file filename
[-path_group groupname_list]
```

This command creates a target-based optimization (TBOpt) file for use with post-route targeted optimization. Once created, this file is passed to the `opt_target_based_opt_file` attribute. You can either provide the command a timing report in a machine-readable format by using the `-gtd_file` parameter or let the command generate the timing report internally by using the values specified for the `-min_slack`, `max_slack`, and -max_paths parameters or the internal default values for these parameters.

Based on this timing report, a bottleneck analysis is performed and the critical instances are stored in a file that is specified using the `-out_file` filename parameter. This file uses the TBOpt format with the INSTS token. The bottleneck analysis computes the number of times an instance appears (on only the failing paths by default) and creates "n" number of path categories for the instances with the highest number of occurrences. Note that "n" is the number you specify using the `-bottleneck_instance_number` parameter of this command.

Also note that by default, multiple instances per path are allowed. This means that if there are multiple critical instances on a path, all of them will be returned, therefore, leading to the generation of the most accurate list of critical instances.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `write_target_based_opt_file` parameter. For a detailed description of the command and all of its parameters, use the man command: `man write_target_based_opt_file`.
`-bottleneck_instance_number integer`	

	Specifies the number of critical bottleneck instances to be generated. This is a required parameter. *Data_type*: int, required

`-early`	Specifies that the TBOpt file will be created based on hold timing's critical nodes. This parameter can be specified along with the following commands to generate and use a hold target file for hold TBOpt. For example:
	`write_target_based_opt_file -early -bottleneck_instance_number 5 -out_file TBOpt_bottleneck_5_hold.txt`
	`set_db opt_target_based_opt_hold_file TBOpt_bottleneck_5_hold.txt`
	`opt_design -post_route -targeted -hold`
	Data_type: bool, optional

`-gtd_file` *filename*	Specifies the name of the machine-readable timing report on which you want to base the critical instance analysis. If this parameter is not specified, the command generates its own file with the `-max_slack` value of `0.0` and `-max_paths` value of `100,000`. *Data_type*: string, optional
`-max_paths` *integer*	Specifies the number of worst paths to be considered, regardless of the endpoint. By default, the value is `100,000`. *Data_type*: int, optional
`-max_slack` *float*	Specifies that only those paths should be considered that have a slack equal to or less than the specified float value in `ns`. By default, the value is `0.0` to ensure that only paths with negative slacks are considered. *Data_type*: float, optional
`-min_slack` *float*	Specifies that only those paths should be considered that have a slack greater than the specified float value in `ns`. *Data_type*: float, optional

`-out_file` *`filename`*	Specifies the name of the TBOpt file that will be generated. This is a required parameter. *Data_type*: string, required
`-path_group` *`groupname_list`*	
	Specifies that only those paths should be considered that are from the specified path groups on which you want to base the critical instance analysis. *Data_type*: string, optional

Example

- The following command generates a TBOpt file with five critical instances using the default values for `-min_slack`, `-max_slack`, and `-max_paths` that will be written to the file, `TBOpt_bottleneck_5.txt`:

  ```
  write_target_based_opt_file -bottleneck_instance_number 5 -out_file
  TBOpt_bottleneck_5.txt
  ```

  The `TBOpt_bottleneck_5.txt` file will contain five critical instances with the appropriate syntax for target-based postRoute optimization.
  For example,

  ```
  INSTS : {mod_i/i_mod_glm_cluster_0/U586} {mod_i/i_mod_glm_cluster_2/U535}
  {mod_i/i_mod_glm_cluster_1/U596} {mod_i/ser_en/U92} {mod_i/ser_en/U101}
  ```

- The following command generates a TBOpt file, `usergenerated.mtarpt` with three critical instances based on a machine-readable timing report generated using the `-machine_readable` parameter of the `report_timing` command:

  ```
  write_target_based_opt_file -bottleneck_instance_number 3 -out_file
  TBOpt_bottleneck_3.txt -gtd_file usergenerated.mtarpt
  ```

- The following command generates a TBOpt file with six critical instances based on only the `reg2reg` path group with the maximum number of worst paths as 3000:

  ```
  write_target_based_opt_file -bottleneck_instance_number 6 -out_file
  TBOpt_bottleneck_6.txt -path_group reg2reg -max_paths 3000
  ```

- The following command generates a TBOpt file with nine critical instances based on a timing analysis of the worst 10,000 paths between `+0.025ns` and `-1.0ns`:

  ```
  write_target_based_opt_file -bottleneck_instance_number 9 -out_file
  TBOpt_bottleneck_9.txt -min_slack -1.0 -max_slack +0.025 -max_paths 10000
  ```

Related Information

- `opt_design`
- The following section in Optimizing Timing chapter of *Innovus Stylus Common UI User Guide*
  - Performing Target-Based Post-Route Optimization

43

TSV Design Commands

- add_obj_on_bump
- add_tsv
- assign_tsv
- check_stacked_die
- create_stress_rule
- delete_tsv
- read_bump_locations
- read_codesign_die_abstract
- read_die_abstract
- read_tsv_config
- unassign_tsv
- write_bump_locations
- write_codesign_die_abstract
- write_die_abstract
- write_micro_bump_mapping_file
- write_spef_between_tsv_dies
- write_design_stack_config
- read_design_stack_config

add_obj_on_bump

```
add_obj_on_bump
[-help]
{{-cell bump_cell_name | -selected | -feedthru } {-relative_pin_layer pin_layer_name | -rect {x1
y1 x2 y2} | -polygon {{x1 y1} {x2 y2}...}}}
[-check_only bump_name | -write_script filename]
{[-shape layer+] [-placement_blockage] [-routing_blockage layer+] [-physical_pin layer+]}
[-shape layer+ [-check_only bump_name] [-incremental][-shape_keep_net ]]
[-feedthru [-cdl_map_file cdl_map_file_name]]
```

Adds shapes on the TSV bumps. You can use this command to add shapes on all the specified TSV bumps or the specified bump. The shape size could be user-defined or same as bump pin on the specified layer.

Parameters

`-help`	Prints a brief description of the `add_obj_on_bump` command usage. For a detailed description of the command, use the `man` command: `man add_obj_on_bump`
`-cdl_map_file` *cdl_map_file_name*	
	Specifies the output cdl map file for LVS. *Data_type*: string, optional
`-cell` *bump_cell_name*	Specifies the name of the bump cell on which the shape is to be added. *Data_type*: string, optional
`-check_only` *bump_name*	Checks if there is a shape on the specified bump and outputs a report file. The `-check_only` option cannot be used with `-write_script`. ⚠ The `-check_only` option can be used only when `-shape` is specified. *Data_type*: string, optional
`-feedthru`	Adds shapes on the bump that contains the TSV layer and whose net is a feedthrough net. *Data_type*: bool, optional

`-incremental`	Adds shapes in the incremental mode. It deletes the shapes without a bump, and adds new shapes on the bumps without shapes. The -incremental parameter only checks the shapes on the same layer.
	In the following sequence of commands, the first command is `add_obj_on_bump -shape M1` and the second command is `add_obj_on_bump -shape M2 -incremental`. In these commands the layers specified are different. Therefore, `add_obj_on_bump` will not delete previously added shapes on `M1`. Also, the result of the command `add_obj_on_bump -shape M2 -incremental` is the same as `add_obj_on_bump -shape M2`.
	`>add_obj_on_bump -shape M1`
	`>add_obj_on_bump -shape M2 -incremental`
	⚠ The `-incremental` option can be used only when `-shape` is specified.
	Data_type: bool, optional

`-shape` *layer+*	Specifies the layer to add new shape. *Data_type*: (layer)+, optional
`-placement_blockage`	Adds a placement blockage on bumps. Only orthogonal polygons are supported. *Data_type*: bool, optional
`-physical_pin` *layer+*	Specifies the layer(s) on which to add physical pins. For a single-pin bump, the name of the physical pin is same as that of the target bump. For a multiple-pin bump, the name of the physical pin is same as that of the bump pin on the same layer. If no bump pin exists on the layer specified for adding the physical pin, the tool chooses the name of any of the pins on the target bump as the pin name of the physical pin. Polygons with non-45-degree edges are supported.
`-polygon {{x1 y1} {x2 y2}...`	
	Specifies the point list of the polygon to add shapes. The center of the specified polygon bounding box aligns with the center of the specified bump cell. *Data_type*: (point)+, optional

`-rect {x1 y1 x2 y2}`	Specifies the rectangle to add shapes. The center of specified rectangle aligns with the center of the specified bump cell. *Data_type*: rect, optional
`-relative_pin_layer` *pin_layer_name*	
	Specifies the pin layer of the pin shape of the specified bump. Adds shapes based on the specified pin shape of the bump. If the `-relative_pin_layer` option is specified while adding objects, the shape of the added object is same as the shape of the relative pin. *Data_type*: (layer), optional
`-routing_blockage` *layer+*	
	Specifies the layer(s) on which to add routing blockages on bumps. Only orthogonal polygons are supported. *Data_type*: (layer)+, optional
`-selected`	Adds shapes on the selected bump. *Data_type*: bool, optional
`-shape_keep_net`	Keeps the net when adding the layer shape. When you specify the `-incremental` option, the tool adds shape with net name if there is no shape with this net on the bump. *Data_type*: bool, optional
`-write_script` *filename*	Specifies the file name to dump a Tcl script for adding objects on bumps. If this option is specified, the tool will not add objects on bumps directly. The `-write_script` option cannot be used with `-check_only`. *Data_type*: string, optional

Example

- The following command adds a shape on layer M1 at the position of the TSV bump whose net is a feedthrough net. The shape on layer M1 is the same as the bump pin shape on layer TSV.

```
add_obj_on_bump -shape M1 -feedthru -relative_pin_layer TSV
```

add_tsv

```
add_tsv
[-help]
[-no_route_blockage]
{[-front_bump  bumpCellName] [-back_bump  bumpCellName] [-tsv_via_name viaName]}
[-front_bump  bumpCellName [-location_type {geometry_center|cell_center} -prefix prefix] [-
stack_via_under_bump stackViaName]]
[-back_bump bumpCellName  [-location_type {geometry_center|cell_center} -prefix prefix]]
[-tsv_via_name viaName [-no_place_blockage  | -spacing s] [[-tsv_feedthru {x y}
     [-stack_via_top_layer layer]] | [-stripe_net pgnetName -stripe_layer layerNum |
     [-stack_via_name stackViaName[-stack_via_marker_layer markerLayerName] -stack_via_array {x
y} -stack_via_pitch {x y}]]]]
{-lower_left_location x y}
[-relative_bump_cell bumpCell | -relative_selected_bump ]} {[-pitch_x_y {x y}] [-
relative_bump_cell bumpCell | -relative_selected_bump ]}
{-upper_right_location x y | -num_x_y x y}
[-relative_bump_cell bumpCell | -relative_selected_bump ]} [{-relative_bump_cell bumpCell | -
relative_selected_bump } [-relative_min_pitch minPitch] [-relative_offset {x y}] [-
relative_bump_type {signal power ground}]]
[-rings n] [-stagger ]
```

Adds TSVs, frontside bumps, and backside bumps under specified conditions. Use this command to create TSVs and bumps after the initial floorplan is loaded.

The command also checks for dummy bump overlaps by default. It checks overlaps between:

- Dummy bumps by checking the OBS overlap.
- Dummy bump and bump by checking the OBS and bump pin overlap.

In case there is dummy bump overlaps, the tool gives a warning message.

Parameters

`-tsv_via_name viaName`

	Specifies the name for TSV via defined in the lef file. If this option is not specified, no TSV will be created. The following arguments are used only with `tsv_via_name`: • `no_place_blockage` • `spacing` • `tsv_feedthru` • `stack_via_top_layer` • `stripe_net` • `stripe_layer` *Data_type*: string, optional
`-front_bump` *bumpCellName*	
	Specifies the name for the frontside bump cell. If this option is not specified, no frontside bump will be created. *Data_type*: string, optional
`-back_bump` *bumpCellName*	
	Specifies the name for the backside bump cell. If this option is not specified, no backside bump will be created. *Data_type*: string, optional
`-location_type {geometry_center cell_center}`	
	Specifies the location type for bump creation. *Data_type*: enum, optional
`-lower_left_location x y`	
	Specifies (in microns) the lower left coordinate of the box in which TSV/Bumps will be created. *Data_type*: point, required
`-num_x_y` x y	Specifies (in microns) the count of the TSV/bump in the horizontal and vertical direction. *Data_type*: point, optional

`-relative_bump_cell` *bumpCell*	Creates TSV or bumps by default at the geometry center of bumps with specified bump cell.
`-relative_bump_type {signal power ground}`	Creates the relative bump type, TSV, or bumps by default at the geometry center of bumps with the specified bump type.
`-relative_min_pitch` *minPitch*	Specifies the minimum pitch between the created bumps/TSVs. The pitch between all the created bumps/TSV should be larger than specified min pitch. The `-relative_min_pitch` option is optional and can be used only if `-relative_bump_cell/-relative_selected_bump` is specified.
`-relative_offset {x y}`	Specifies the offset of created bump/TSV location compared with the relative bump. The geometry center of the created bump/TSV is the geometry center of the relative bumps plus the offset. `-relative_offset` is optional and can be used only if `-relative_bump_cell`/`-relative_selected_bump` is specified.
`-relative_selected_bump`	Creates TSV or bumps by default at the geometry center of the selected bumps.

`-rings` *n*	Creates TSV along rings on the perimeter. If `-rings` is specified, *n* defines the number of rings.
	Data_type: int, optional
	You can create TSV in the following ways:
	• Define the lower left and upper right location of the ring by -`lower_left_location` and `-upper_right_location`, and then the pitch value of x and y direction by -`pitch_x_y`. For example, the following command creates TSV in which the lower-left is at `{x1 y1}` and the upper-right is at `{x2 y2}`
	`add_tsv -lower_left_location {x1 y1} -upper_right_location {x2 y2} -pitch_x_y {px py} -stagger -rings 1`
	`*********`
	`*  *`
	`*  *`

	 • Define the lower left location of the ring by `-lower_left_location`, and then specify the number and pitch value of tsv in x and y directions by `-num_x_y` and `-pitch_x_y`, respectively. For example, the following command creates TSV in which the lower left is at {x1 y1} `-lower_left_location {x1 y1} -num_x_y {12 8} -pitch_x_y {px py} -rings 2`
`-pitch_x_y` *x y*	Specifies (in microns) the distance between TSV centers in the x and y direction. *Data_type*: point, required
`-prefix` *prefix*	Adds a prefix name for bump during bump creation and bump mapping. *Data_type*: string, optional
`-no_place_blockage`	The placement blockage overlapped with TSV cut will be added as default if `-no_place_blockage` is not specified. *Data_type*: bool, optional

`-no_route_blockage`	The routing blockage for each crossing layer will be added if `-noRouteBlkg` is not specified. *Data_type*: bool, optional
`-tsv_feedthru` *`sizex sizey`*	
	Creates the feedthru. `sizeX` and `sizeY` are the dimensions, in microns, for the stack vias aligned with the TSV. *Data_type*: point, optional **Note:** Feedtrhu is a collection of components including TSV, the stacked via (from the first metal to the top metal) aligned with the TSV, and two bumps on the frontside and backside.
`-stack_via_marker_layer` *`markerLayerName`*	Specifies the mark layer name. If cut layers of stack via cell are different, you can use `[-stack_via_marker_layer {`*`markLayerName1 markLayerName2`*`}]` to define different mark layers *Data_type*: string, optional
`-stack_via_name` *`stackViaName`*	Specifies the stack via cell name. The tool automatically adds the specified via cell stacking to the top layer of long TSV. You can add TSV stack vias on both the front-side and back-side layers using `-stack_via_name {`*`stackViaName1 stackViaName2`*`}` *Data_type*: string, optional
`-stack_via_top_layer` *`layer`*	
	Defines the top layer that the TSV structure should connect to. For example, if you specify M5 for this parameter, Innovus will create a structure like `TSV+VIA12+M2+VIA23+M3+…..+M5`. This parameter must be used with `-tsv_feedthru`. *Data_type*: string, optional
`-stack_via_under_bump` *`stackViaName`*	

	Specifies the stack via name under the front bump. *Data_type*: string, optional The tool automatically adds the specified via cell stacking to the bottom layer of front-side bump.
`-stack_via_array {x y}`	Specifies the number of columns and rows for the stack via array. The `-stack_via_array` and `-stack_via_pitch` options can only be specified when `-stack_via_name` is specified. `-stack_via_array` and `-stack_via_pitch` are specified at the same time. The value of `-stack_via_pitch` should be larger than stack via size. Otherwise, the stack via will overlap each other and cause DRC. You can also define different arrays when you add TSV stack via on both the frontside and backside layers with `-stack_via_array {{x1 y1} {x2 y2}}`. If there are two stack vias and only one needs to be defined, you can use `-stack_via_array {{x1 y1} {1 1}}`.
`-stack_via_pitch {x y}`	Specifies the pitch_x and pitch_y of the stack via. The `-stack_via_array` and `-stack_via_pitch` options can only be specified when `-stack_via_name` is specified. `-stack_via_array` and `-stack_via_pitch` are specified at the same time. The value of `-stack_via_pitch` should be larger than stack via size. Otherwise, the stack via will overlap each other and cause DRC. You can also define different pitches when you add TSV stack via on both the frontside and backside layers with `-stack_via_pitch {{x1 y1} {x2 y2}}`

`-stagger`	Staggers (offsets) the TSV/bump array.
	Data_type: bool, optional
	For example:
	without `-stagger` with `-stagger`
	`*****` `*****`
	`*****` `****`
	`*****` `*****`
	`*****` `****`
	`*****` `*****`

`-stripe_net` *pgnetName*	
	Declares that the TSV will be added beneath the power stripe, a stack via will be generated to touch the power stripe, and the new added TSV will be assigned by the power or ground net of the power stripe. *Data_type*: **Note:** This options should be used along with the `-stripe_layer` option. *Data_type*: string, optional
`-stripe_layer` *layerNum*	
	Specifies the layer number or layer name for the stripe net when adding TSV under stripe net. *Data_type*: string, optional **Note:** This option should be used along with `-stripe_net` option.
`-spacing` s	Specifies the placement blockage size. Blockage size = TSV cut size + 2*s. If this option is not specified, "s" will be with the value defined in LAYER TSV session in LEF file. *Data_type*: float, optional
`-upper_right_location x y`	

	Specifies (in microns) the upper right coordinate of the box in which TSV/Bumps will be created. *Data_type*: point, optional

⚠ At least one option of the following arguments is required for the `add_tsv` command:

- `tsv_via_name`
- `front_bump`
- `back_bump`

Examples

- The following command creates a 10 x 10 TSV array with a pitch of 40 microns and the lower left at (300 300):
  ```
  add_tsv -lower_left_location 300 300 -num_x_y 10 10 -pitch_x_y 40 40 -tsv_via_name TSVVIA
  ```
- The following command creates a backside array in box (300 300 700 700) with a pitch of 40 microns:
  ```
  add_tsv -lower_left_location 300 300 -upper_right_location 700 700 -pitch_x_y 40 40 -
  back_bump BUMPCELL
  ```

Related Information

- `assign_tsv`
- `delete_tsv`
- `read_tsv_config`
- Design Methodology for 3D IC with Through Silicon Via chapter in *Innovus Stylus Common UI User Guide*.
- Tools Menu - TSV in *Innovus Stylus Common UI Menu Reference*.

assign_tsv

```
assign_tsv
[-help]
[-write_down_level_bump]
[-selected_to_net netName | [[-signal | -pg_pad] [-feedthru | -non_feedthru] [-net netlist | -exclude_nets excludeNetList] [-regions {x1 y1 x2 y2 ...] [-exclude_region x1 y1 x2 y2 ...] [-tsv_via_name tsvViaCellName] [-interposer ] ] ]
[{[-back_bump] [-front_bump]} [-force_current_die ]]
```

Assigns nets to TSVs and/or frontside and backside bumps, and/or feedthrus. The command considers all dies in the 3D IC System by using the die abstract files, and optimizes bump assignment for the global nets which connect these dies. If you run the `read_die_abstract` command to import die abstract files of other chips inside the 3D IC System before running the `assign_tsv` command, `assign_tsv` will optimize TSV/bump assignment by considering the whole system.

Note: Use this command after the TSVs and bumps are created and the instances connected with the IO pins are placed.

Parameters

`-help`	Print out the command usage.
`-back_bump`	Declares that the back side bumps and the TSVs will be assigned automatically. *Data_type*: bool, optional
`-force_current_die`	Assigns all IO pins to backside bumps or frontside bump directly without checking the XML file and the top level netlist. You can use `-force_current_die` only if `-back_bump` or `-front_bump` is specified. *Data_type*: bool, optional
`-front_bump`	Declares that the front side bumps will be assigned with the IO pins automatically. *Data_type*: bool, optional
`-tsv_via_name tsvViaCellName`	
	If this option is specified, only the TSV with the VIA cell `tsvViaCellName` will be assigned. Others will be ignored. *Data_type*: string, optional

`-selected_to_net` *netName*	
	Assigns the selected TSVs and bumps with a specified net. *Data_type*: string, optional **Note:** If the selected TSV belongs to a feedthru, the whole feedthru will be assigned with the net. This option is used for manual assignment and is exclusive to any other options for auto assignment.
`-signal`	Declares that only signal will be assigned. *Data_type*: bool, optional
`-pg_pad`	Declares that only power/ground net will be assigned. *Data_type*: bool, optional
`-feedthru`	Declares that the command assigns only the feedthru net. *Data_type*: bool, optional
`-non_feedthru`	Declares that the command assigns only the nonfeedthru signal or power/ground net. *Data_type*: bool, optional
`-nets` *netlist*	Assigns the specified nets to the near TSV/bumps. *Data_type*: string, optional
`-exclude_nets` *excludenetlist*	
	Excludes the nets in the specified boxes. *Data_type*: string, optional
`-regions {`*x1 y1 x2 y2 ...*`}`	
	Assigns the TSV/bump within the box only. *Data_type*: double, optional
`-exclude_regions {`*x1 y1 x2 y2 ...*`}`	
	Excludes the TSV/bumps in the specified boxes. *Data_type*: double, optional
`-interposer`	Declares that the assignment is done in silicon interposer without any instance. *Data_type*: bool, optional

`-write_down_level_bump`	Writes down the level bump file for each hierarchical instance. *Data_type*: bool, optional

Notes:

- If neither `-signal` nor `-pg_pad` is specified, both signal and pgpad will be assigned.
- If neither `-feedthru` nor `-non_feedthru` is specified, both feedthru and non-feedthru net will be assigned.
- If neither `-front_bump` nor `-back_bump` is selected, none of the bumps will be assigned. However, the TSV will still be assigned.

Examples

- The following command assigns the signal IO pins to the frontside bump:
  ```
  assign_tsv -front_bump -signal
  ```
- The following command assigns the selected TSV with net VDD:
  ```
  addTSV -selected_to_net VDD
  ```

Related Information

- `delete_tsv`
- `read_die_abstract`
- Design Methodology for 3D IC with Through Silicon Via chapter in *Innovus Stylus Common UI User Guide*.
- Tools Menu - TSV in *Innovus Stylus Common UI Menu Reference*.

check_stacked_die

```
check_stacked_die
[-help]
[-check_type type_name]
[-chip_names {chip1 chip2 ...}]
```

Checks die edge violation between adjacent dies. Once you define an OBS layer inside a macro, the adjacent die edge should not overlap this OBS area. If there is any violation between macro OBS and adjacent die edge, it will be marked in the GUI window.

This command must be used after `read_die_abstract` to input adjacent die's die abstract file.

Parameters

`-help`	Prints a brief description of the `check_stacked_die` command usage. For a detailed description of the command, use the `man` command: `man check_stacked_die`
`-check_type type_name`	Specifies the check constrain types. Currently, only the `die_edge` and `bump_keepout` zone checking are supported. *Data_type*: string, optional
`-chip_names {chip1 chip2 ...}`	Specifies the name of the adjacent chip whose edge is to be checked for violation. In a 3D-IC design, there could be more than one top die (in interposer design), so you can specify the chip that needs to be checked. Using this parameter, you can check one or more chips at the same time. *Data_type*: string, optional

Examples

- The following command checks die edge violation between the macro's keep-out area and adjacent die `mother_die`:

  ```
  check_stacked_die -chip_names {mother_die}
  ```

Related Information

- `read_die_abstract`

create_stress_rule

```
create_stress_rule
[-help]
{ {-layer layerName -spacing_table spacing_table [-chip_name chipName]} | [-reset ] }
```

Defines a spacing table for spacing rules. Since the spacing rule varies with the location of the keep out zone in the chip, user can define different spacing tables for different spacing rule. These spacing tables are used by the tool to identify different scenarios and select the correct spacing constraint for a scenario.

Parameters

`-help`	Prints a brief description of the `create_stress_rule` command usage. For a detailed description of the command, use the `man` command: `man create_stress_rule`
`-chip_name` *`chipName`*	Specifies the name of the chip for which the spacing table is used for checking violation. If this option is not specified, the tool applies the spacing rule for all the chips above, below, and current based on the masterslice layer definition in the LEF file *Data_type*: string, optional
`-layer` *`layerName`*	Specifies the name of the stress layer for which the spacing table is used for checking violation. *Data_type*: string, optional
`-reset`	Resets all stress rules. *Data_type*: bool, optional
`-spacing_table` *`spacing_table`*	Specifies the spacing table definition, {x1 y1} {x2 y2}. *Data_type*: string, optional

delete_tsv

```
delete_tsv
{-all | -selected| -unassigned}
```

Deletes the specified TSVs. If the TSV belongs to a feedthru, the entire feedthru will also be deleted.

Parameters

`-all`	Deletes all the TSVs. *Data_type*: bool, required
`-selected`	Deletes the selected TSVs. *Data_type*: bool, required
`-unassigned`	Deletes the unassigned TSVs. *Data_type*: bool, required

Examples

- The following command deletes all the TSVs:
  ```
  delete_tsv -all
  ```

Related Information

- `add_tsv`
- `assign_tsv`
- `unassign_tsv`

read_bump_locations

```
read_bump_locations
-in_file fileName
[-allow_overlap_control {keep_all keep_existing_bumps keep_new_bumps}]
[-die_box]
[-force_current_die]
[-ignore_route_blockages]
[{-back_bump_cell bumpCellName | -bump_cell_from_bump_file } [-rotate_bump  -
prefix prefixName]]
[{-front_bump_cell bumpCellName | -bump_cell_from_bump_file } [-rotate_bump  -prefix prefixName]
[-stack_via_under_bump stackViaName]]
[-prefix prefixName | -bump_name_from_bump_file ]
[-check_alignment ] [
-non_feedthru ]
[-feedthru ]
[-tsv_master viaCellName
  [-feedthru_size {x y}]
  [-offset {x y}]]
```

The command creates or assigns objects on the current die according to the bumps on the adjacent die. In a TSV design, the micro bumps on the adjacent die should be aligned, and the aligned micro bumps should be with the same signal. Specify this command to sync-up the micro bump between dies. Firstly, the command tries to find the front side bump/back side bump/TSV at the location aligned to the bump on the adjacent dies. If all the specified bumps/TSVs are found on the current die, the command will assign the found bump/TSV with the same signal as the bump on the adjacent die; if none of the specified bumps/TSV are found, the command will create the specified bump/TSV with the same signal at the aligned position.

This command supports the combo bump and embedded bump flows.

Use this command after reading the stacked die config file (XML), by using `read_tsv_config`, as the die configuration is required for the bump alignment.

The command also checks for dummy bump overlaps by default. It checks overlaps between:

- Dummy bumps by checking the OBS overlap.
- Dummy bump and bump by checking the OBS and bump pin overlap.

In case there is dummy bump overlaps, the tool gives a warning message.

Note: A combo bump is a group of bumps that are treated as one bump, which means all bumps in the same combo bump will be created, assigned, unassigned, and manipulated at the same time. This command synchronizes the combo bump information between adjacent chips automatically. All bumps belonging to the same combo bump will be treated as one unit. Embedded bump is the bump at the block-level, and the information of these bumps can be exchanged between chips at the top-level.

Parameters

`-allow_overlap_control {keep_all keep_existing_bumps keep_new_bumps}`	Specifies the overlap control type. The allowed values for the `-allow_overlap_control` parameter are: • `keep_all`: Keeps all overlapping bumps. • `keep_existing_bumps`: Keeps the existing bumps before calling `read_bump_locations` if there is an overlap. If this option is specified, `read_bump_locations` keeps creating a bump if there is an overlap and continues creating the remaining bumps. A warning message is displayed when bump creation is skipped to avoid overlap. • `keep_new_bumps`: Removes the existing bumps before calling `read_bump_locations`, places the newly created bump if there is overlap, and then continues creating the remaining bumps. A warning message is displayed when deleting the existing bumps to avoid overlap. If the status of the existing bumps to be deleted is covered or fixed, a warning is displayed and the new bumps overlapping with the covered or fixed old bump are not kept.
`-back_bump_cell` *`bumpCellName`*	Creates or assigns the backside bumps at the location aligned with the bumps on the adjacent die. *Data_type*: string, optional
`-bump_cell_from_bump_file`	Creates bumps with the same bump cell defined in the bump file. *Data_type*: bool, optional
`-bump_name_from_bump_file`	Creates bumps with the same bump name defined in bump file. *Data_type*: bool, optional
`-check_alignment`	Use this parameter to check the alignment between current die and the adjacent dies. It also checks the overlap between the OBS layer and bumps (both front and back bumps). When this parameter is specified, all other parameters are ineffective. *Data_type*: bool, optional
`-die_box`	Draws the die bounding box based on the DieBox value in the bump file. *Data_type*: bool, optional
`-feedthru`	Creates or assigns the bump with feedthru net only. If `-feedthru_size` and `-tsv_master` are specified, the TSV and the stacked via of the feedthru will also be created. *Data_type*: bool, optional

`-feedthru_size` *sizex sizey*	
	Specifies the size for feedthru. This option is exclusive with the `-non_feedthru` option. *Data_type*: point, optional
`-force_current_die`	Specifies to consider the bump file as the current die bump file. Skip to check if the specified bump file is for current die or adjacent die or a block.
`-front_bump_cell` *bumpCellName*	
	Creates or assigns the frontside bumps at the location aligned with the bumps on the adjacent die. *Data_type*: string, optional
`-ignore_route_blockages`	Allows creation of bumps on route blockages. When this option is specified, the tool ignores checking route blockages when creating bumps. *Data_type*: bool, optional
`-in_file` *fileName*	Specifies the name of the bump location file of the adjacent die. *Data_type*: string, required
`-non_feedthru`	Creates or assigns the bump with normal net only. *Data_type*: bool, optional
`-offset x y`	Specifies the offset from the TSV to the corresponding bump. *Data_type*: point, optional **Note:** You can also specify the offset in the bump file by appending `-offset x y` to the line. The offset in the bump file has higher priority than the offset specified in the command line. If `-offset` is specified at both command line and bump file, the program will follow the offset specified in the bump file.
`-prefix` *prefixName*	Adds a prefix name for bump during bump creation and bump mapping. *Data_type*: string, optional The default name of bumps are `frontBump_*` or `backBump_*` with serial numbers. If `-prefix` *prefixName* is applied, created bump names are `prefixName_frontBump_*` or `prefixName_backBump_*`.

`-rotate_bump`	Rotates bumps during bump mapping according to the chip and bump orientation information of the adjacent die. *Data_type*: bool, optional
`-stack_via_under_bump` *stackViaName*	Specifies the stack via name under the front bump. The tool automatically adds the specified via cell stacking to the bottom layer of front-side bump. *Data_type*: string, optional
`-tsv_master` *viaCellName*	Creates or assigns the TSV at the location aligned (or with offset if `-offset` is specified) to the bumps on the adjacent die. *Data_type*: string, optional

Note: If neither `-feedthru` nor `-non_feedthru` is specified, the bump/TSV with both feedthru net and non-feedthru net will be created and assigned.

Examples

- The following command reads a bump file and creates the corresponding frontside bump on current die:

  ```
  read_bump_locations bumpfile -front_bump_cell front_bump_cell
  ```

- The following command reads a bump file and creates the corresponding backside bump and TSV current die:

  ```
  read_bump_locations bumpfile -back_bump_cell back_bump_cell -tsv_master tsv_via
  ```

Related Information

- `write_bump_locations`
- `read_tsv_config`
- Design Methodology for 3D IC with Through Silicon Via chapter in *Innovus Stylus Common UI User Guide*.

read_codesign_die_abstract

```
read_codesign_die_abstract
[-help]
file_name
[-check_only]
[-no_different_views]
```

Reads the die abstract directly to update Innovus. The command provides flexibility in terms of what is read and the elimination of the I/O Update file and TCL code.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `read_codesign_die_abstract` parameter. For a detailed description of the command and all of its parameters, use the man command: `man read_codesign_die_abstract`
`file_name`	Specifies the name of the die abstract file.
`-check_only`	This is only valid with the `-no_different_views` option. If these two options are given, then the difference viewer will not be used, and the Innovus db will not be updated.
`-no_different_views`	When the `-no_different_views` option is used, the difference viewer will not be utilized. The die abstract will be read and the Innovus db will be updated with all the changes unless the `-check_only` option is provided.

Related Information

- `write_codesign_die_abstract`

read_die_abstract

```
read_die_abstract
DIEABSTRACTFILE
```

Reads the die abstract of the adjacent die. With this command, the abstract of the adjacent die is imported and displayed overlapped with the current design.

This command supports the embedded bump flow for hierarchical designs.

The `read_die_abstract` command is used after reading the stacked die config file (XML) by using `read_tsv_config` because the die configuration is required for die alignment. You can turn off the import adjacent die objects by changing the visibility of abs* layers in the *Custom* tab of the *Color Preferences* form.

Parameters

DIEABSTRACTFILE	Specifies the name of the die abstract file. *Data_type*: string, required

Examples

- The following command reads a die abstract file `die1.abs`:

  ```
  read_die_abstract die1.abs
  ```

Related Information

- `read_tsv_config`
- `write_die_abstract`
- The *Color Preferences* section of the The Main Window chapter in *Innovus Stylus Common UI Menu Reference*.

read_tsv_config

```
read_tsv_config
{[-stacked_die_file filename] [-power_file filename]}
[-chip_name chipName]
[-top_netlist topLevelNetlist ]
```

Imports 3D IC configuration files of the whole design, including the stack die configuration file, power net configuration file and top-level netlist.

Parameters

-stacked_die_file filename	
	Reads the stack die configuration file. *Data_type*: string, optional
-power_file *filename*	Imports the power net configuration file. *Data_type*: string, optional
-chip_name *name*	Specifies the hierarchical instance name (chip) name of the current design in the top-level netlist. *Data_type*: string, optional
-top_netlist *topLevelNetlist*	Imports the top level netlist to support 3D IC design in OA mode. *Data_type*: string, optional

Examples

- The following command reads the stacked die configuration file and power configuration file and specifies that the current design is named `mem_1`:

  ```
  read_tsv_config -stacked_die_file stacked_die.xml -power_file power.side -chip_name mem_1
  ```

- Below is the example of a stack die configuration file:

```
<StackChip>
  <TopDesign name="top" />
  <Tier number=1>
    <Chip name="i_asic_entity" orientation="R0" faceUp="Yes" llx=0 lly=0 sizeX=6000.0
sizeY=6000.0 scaleFactor=1 />
  </Tier>
  <Tier number=2>
```

```
    <Chip name="i_ramextra" orientation="R0" faceUp="Yes" llx=1500.0 lly=1250.0 sizeX=3500.0
sizeY=3000.0 scaleFactor=1 />
  </Tier>
</StackChip>
```

- Below is the example of a power net configuration file:

```
VDD i_asic_entity   sharable

VDD i_ramextra       backside

VSS i_asic_entity   sharable

VSS i_ramextra      backside

Alias:

i_asic_entity/VDD i_ramextra/VDD

i_asic_entity/VSS i_ramextra/VSS
```

Related Information

- Design Methodology for 3D IC with Through Silicon Via chapter in *Innovus User Guide*.
  - Stack Configuration Input
  - Power Connectivity Input

unassign_tsv

```
unassign_tsv
{-net_name netname| -all |-signal | -power | -ground | -selected}
```

Unassigns the specified TSV. This command is used for manual operating on TSV or iteration between TSV assignment and instance placement.

Parameters

`-net_name` *netname*	Unassigns the specified net from TSVs. *Data_type*: string, required
`-all`	Unassigns all the TSVs. *Data_type*: bool, required
`-signal`	Unassigns the signal net from TSVs. *Data_type*: bool, required
`-power`	Unassigns the power net from TSVs. *Data_type*: bool, required
`-ground`	Unassigns the ground net from TSVs. *Data_type*: bool, required
`-selected`	Unassigns selected TSVs. *Data_type*: bool, required

Examples

- The following command reads unassign all the TSVs with power net:
  ```
  unassign_tsv -power
  ```

Related Information

- `add_tsv`
- `assign_tsv`
- Design Methodology for 3D IC with Through Silicon Via chapter in *Innovus Stylus Common UI User Guide*.

write_bump_locations

```
write_bump_locations
out_file
[-selected]
```

Writes the bump information to a file. It also writes the bump keep-out area information to the bump file. Specify this command after the bumps are created and/or assigned.

This command supports the combo bump and embedded bump flows.

Parameters

out_file	Specifies the file name. *Data_type*: string, required
-selected	Specifies to output information related to selected bumps to the bump file. Select the required bumps in the GUI or on the command line, and specify this parameter to output the bump information. You can use this parameter when you want to output bump information that is related to a specific block in hierarchical designs. *Data_type*: bool, optional

Examples

- The following command create a bump information file `die1.bump`:

  ```
  write_bump_locations die1.bump
  ```

- The below example shows a sample a bump information file:

  ```
  === die1.bump ===

        ChipName: ramextra

        DieBox: 0.000000 0.000000 3500.000000 2999.910000

        OBS: BumpKOZ -poly 906.030 1047.445 906.030 100.000 100.000 100.000 100.000 1047.445
  -spacing 0.000 -backSide

        OBS: BumpKOZ -poly 906.030 1952.465 906.030 2899.910 100.000 2899.910 100.000
  1952.465 -spacing 0.000 -backSide

        OBS: BumpKOZ -poly 2593.970 1047.445 2593.970 100.000 3400.000 100.000 3400.000
  1047.445 -spacing 0.000 -backSide

        OBS: BumpKOZ -poly 2593.970 1952.465 2593.970 2899.910 3400.000 2899.910 3400.000
  1952.465 -spacing 0.000 -backSide
  ```

```
Bump: backBump_499 backUBump_40 3450.000 2850.000 -backside
Bump: backBump_498 backUBump_40 3450.000 2650.000 -backside
Bump: backBump_497 backUBump_40 3450.000 2450.000 VSS -ground -backside
...
Bump: backBump_438 backUBump_40 50.000 250.000 VSS -ground -backside
Bump: backBump_437 backUBump_40 50.000 50.000 VDD -power -backside
Bump: backBump_436 backUBump_40 3400.000 1800.000 qa_3[31] -backside
Bump: backBump_435 backUBump_40 3400.000 1700.000 qb_3[31] -backside
...
Note: backUBump_40 is the backBump cell name.
The design "ramextra" origin is the coordinate original of bump information file.
```

- The following command creates the bump information file, `die2.bump`, for a multiple-pin feedthrough bump:

  ```
  write_bump_locations die2.bump
  ```

  The below example shows a sample a bump information file for a multiple-pin feedthrough bump：

  ```
  === die1.bump ===

  ................................
  ................................
  ................................
  ................................
  Bump: Bump_1 uBUMP 200 200 PIN_1 CLK_IN PIN_1 CLK_OUT PIN2
  Bump: Bump_2 uBUMP 200 400 VSS -ground
  ................................
  ................................
  ................................
  ................................
  ................................
  ................................
  ................................
  ................................
  ```

Related Information

- `read_bump_locations`
- Design Methodology for 3D IC with Through Silicon Via chapter in *Innovus Stylus Common UI User Guide*.

write_codesign_die_abstract

```
write_codesign_die_abstract
[-help]
out_file
[-block_pin_layer_bottom layer_name]
[-block_pin_layer_top layer_name]
[-block_pin_min_size x y]
[-block_pin_name_includes pin_name]
[-bump_pin_layer_bottom layer_name]
[-bump_pin_layer_top layer_name]
[-constraints file_name]
[-die_shrink_by num]
[-die_thickness num]
[-include_macro_obstruction]
[-io_macros macro_names]
[-io_pin_layer_bottom layer_name]
[-io_pin_layer_top layer_name]
[-io_pin_min_size pin_size]
[-io_pin_name_includes pin_name]
[-layer_detail]
[-macro_obstruction_layer_bottom layer_name]
[-macro_obstruction_layer_top layer_name]
[-no_block_macros]
[-no_blockages]
[-no_rdl_routing]
[-power_routing]
[-power_routing_layer_bottom layer_name]
[-power_routing_layer_top layer_name]
[-rdl_routing_layer_bottom layer_name]
[-rdl_routing_layer_top layer_name]
[-scribe_all distance_in_microns]
[-scribe_east distance_in_microns]
[-scribe_north distance_in_microns]
[-scribe_south distance_in_microns]
[-scribe_west distance_in_microns]
[-seal_ring_all distance_in_microns]
[-seal_ring_east distance_in_microns]
[-seal_ring_north distance_in_microns]
[-seal_ring_south distance_in_microns]
[-seal_ring_west distance_in_microns]
[-version {4.0}]
```

Writes the abstract of current die to a file. It also writes the OBS for die edge checking to the die abstract file. Specify this command after the initial floorplan is loaded. The file will be imported by the adjacent die and therefore the current design will be honored by the adjacent die. The write_codesign_die_abstract

command is similar to the writeDieAbstract except that the write_codesign_die_abstract command provides options for filtering portions of the abstract which may not be required for some flows. It also provides the option to write out a third-party die abstract for ICs that do not have TSVs.

This command supports the embedded bump flow for hierarchical designs.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `write_codesign_die_abstract` parameter. For a detailed description of the command and all of its parameters, use the man command: `man write_codesign_die_abstract`
`out_file`	Specifies the name of the die abstract file. *Data_type*: string, required
`-block_pin_layer_bottom` *`layer_name`*	Specifies the name of the legal routing layer for block macros. If specified, the layers below the mentioned layer should not be the output to the die abstract. *Data_type*: string, optional
`-block_pin_layer_top` *`layer_name`*	Specifies the name of the legal routing layer for block macros. If given, the layers above the mentioned layer should not be the output to the die abstract. *Data_type*: string, optional
`-block_pin_min_size` *`x y`*	Defines the minimum size of the pin that will be output to the die abstract. *Data_type*: float, optional
`-block_pin_name_includes` *`pin_name`*	Specifies that only pins that contain the mentioned value will be in the output. *Data_type*: string, optional
`-bump_pin_layer_bottom` *`layer_name`*	Specifies the legal routing layers name for bump macros. If given, the layers below the specified layer are not to be output to the die abstract. *Data_type*: string, optional

`-bump_pin_layer_top` *layer_name*	Specifies the legal routing layer name; used for bump macros. If given, the layers above the specified layer are not to be output to the die abstract. *Data_type*: string, optional
`-constraints` *file_name*	Specifies the name of the file that lists the constraints. This file will be read and the constraints will be applied to the design. *Data_type*: string, optional
`-die_shrink_by` *num*	Specifies the percentage by which the die should be shrunk. A typical shrink value is 10 (for a 10% optical shrink). Default: `0` The default value, `0`, indicates no shrink. *Data_type*: float, optional
`-die_thickness` *num*	Specifies the die thickness. Default: `0` *Data_type*: float, optional
`-include_macro_obstruction`	Specifies to give LEF macro OBS shapes in the output to the die abstract. By default, OBS obstructions are not included. *Data_type*: bool, optional
`-io_macros` *macro_names*	Identifies the I/O macros that are to be written to the die abstract. If a macro is not specified, co-design is done at the bump level. *Data_type*: string, optional
`-io_pin_layer_bottom` *layer_name*	Specifies the legal routing layer name; used for I/O macros. If given, the layers below is not to be output to the die abstract. *Data_type*: string, optional
`-io_pin_layer_top` *layer_name*	Specifies the legal routing layers' name; used for I/O macros. If given, the layers above is not to be output to the die abstract. *Data_type*: string, optional

`-io_pin_min_size` *pin_size*	This parameter defines the minimum size of pin that will be output to the die abstract. The pin's x and y dimensions must be at least this value. *Data_type*: float, optional
`-io_pin_name_includes` *pin_name*	This string indicates that only pins that contain the string value will be output. *Data_type*: string, optional
`-layer_detail`	when given, indicates that the layer_details is to be output to the die. *Data_type*: bool, optional
`-macro_obstruction_layer_bottom` *layer_name*	If specified, identifies the bottom layer for which OBS shapes are exported to the die abstract. The default value for this option is the top metal layer. *Data_type*: string, optional
`-macro_obstruction_layer_top` *layer_name*	If specified, identifies the top layer for which OBS shapes are exported to the die abstract. *Data_type*: string, optional
`-no_block_macros`	Indicates that block macros should not be the output to the die abstract. *Data_type*: bool, optional
`-no_blockages`	Indicates that blockages should not be the output to the die abstract. *Data_type*: bool, optional
`-no_rdl_routing`	Indicates that the rdl routing should not be the output to the die abstract. *Data_type*: bool, optional
`-power_routing`	Indicates that the power routing should not be the output to the die abstract. *Data_type*: bool, optional

`-power_routing_layer_bottom` *layer_name*	Specifies the names of the legal routing layers for power routing. If given, the layers below are not to be output to the die abstract. *Data_type*: string, optional
`-power_routing_layer_top` *layer_name*	Specifies the names of legal routing layers for power routing. If given, the layers above are not output to the die abstract. *Data_type*: string, optional
`-rdl_routing_layer_bottom` *layer_name*	Specifies the names of legal routing layers for rdl routing. If given, the layers below are not to be output to the die abstract. *Data_type*: string, optional
`-rdl_routing_layer_top` *layer_name*	Specifies the names of legal routing layers for rdl routing. If given, the layers above are not to be output to the die abstract. *Data_type*: string, optional
`-scribe_all` *distance_in_microns*	Specifies the distance in microns from the die boundary to the edge of the silicon after being cut from the wafer from all sides. *Data_type*: float, optional
`-scribe_east` *distance_in_microns*	Specifies the distance in microns from the die boundary to the edge of the silicon after being cut from the wafer from the East side. *Data_type*: float, optional
`-scribe_north` *distance_in_microns*	Specifies the distance in microns from the die boundary to the edge of the silicon after being cut from the wafer from the North side. *Data_type*: float, optional
`-scribe_south` *distance_in_microns*	Specifies the distance in microns from the die boundary to the edge of the silicon after being cut from the wafer from the South side. *Data_type*: float, optional
`-scribe_west` *distance_in_microns*	Specifies the distance in microns from the die boundary to the edge of the silicon after being cut from the wafer from the West side. *Data_type*: float, optional

`-seal_ring_all` *distance_in_microns*	Species the distance of the seal ring from the die boundary on all sides. Seal ring is a border around the die boundary that is put in pre-shrink. *Data_type*: float, optional
`-seal_ring_east` *distance_in_microns*	Species the distance of the seal ring from the die boundary from the East side. *Data_type*: float, optional
`-seal_ring_north` *distance_in_microns*	Species the distance of the seal ring from the die boundary from the North side. *Data_type*: float, optional
`-seal_ring_south` *distance_in_microns*	Species the distance of the seal ring from the die boundary from the South side. *Data_type*: float, optional
`-seal_ring_west` *distance_in_microns*	Species the distance of the seal ring from the die boundary from the West side. *Data_type*: float, optional
`-version {4.0}`	Specifies the version number of die abstract. *Data_type*: enum, optional

Related Information

- `read_codesign_die_abstract`

write_die_abstract

```
write_die_abstract
out_file
[-no_filter]
```

Writes the abstract of current die to a file. It also writes the OBS for die edge checking to the die abstract file. Specify this command after the initial floorplan is loaded. The file will be imported by the adjacent die and therefore the current design will be honored by the adjacent die.

This command supports the embedded bump flow for hierarchical designs.

Note: By default, the proposed behavior for `write_die_abstract` is:

- For MACRO pin geometries, output ports on top metal layer in all MACROS of all classes.
- For routing, output routing for all top level Verilog ports, and for power and ground, output only those wires with a SHAPE of IOWIRE.

Parameters

out_file	Specifies the file name. *Data_type*: string, required
-no_filter	Turn off to filter the die abstract contents. *Data_type*: bool, optional

Examples

- The following command creates a die abstract file `die1.abs`:

  ```
  write_die_abstract die1.abs
  ```

Related Information

- `read_die_abstract`
- Design Methodology for 3D IC with Through Silicon Via chapter in *Innovus Stylus Common UI User Guide*.

write_micro_bump_mapping_file

```
write_micro_bump_mapping_file
[-help]
bump_map_file
[-merge_with {filename1, filename2,...}]
```

Writes the micro bump mapping file for Quantus and Voltus use. Use this command after the aligned bump pair is created by `read_bump_locations` command. This command supports the embedded bump flow for hierarchical designs.

Parameters

`-help`	Prints a brief description of the `write_micro_bump_mapping_file` command usage. For a detailed description of the command, use the `man` command: `man write_micro_bump_mapping_file`
bump_map_file	Specifies the file name. *Data_type*: string, required
`-merge_with {filename1, filename2,...}`	
	If there are more than two dies in the design, this command may be called more than one time. This option should be specified at the last time to merge the previously generated files. *Data_type*: string, optional

Examples

- The following command creates a bump mapping file:
  `write_micro_bump_mapping_file bump.map`

  Here is an example of a bump mapping file, `bump.map`:

```
############################################################
# Generated by: Cadence Innovus 21.12-e043_1
# OS: Linux x86_64(Host ID noivl-leenap)
# Generated on: Mon Sep 20 11:16:51 2021
# Design: top1
# Command: write_micro_bump_mapping_file bump.map
############################################################
# Format:
# SIGNAL: <Other Die Name>(<Other Die Module Name>) <Other Die Net name> <Other Die Pin
name> <Current Die Name>(<Current Die Module Name>) <Current Die Net name> <Current DiePin
name>
# POWER/GROUND: <Other Die Name>(<Other Die Module Name>) <Other Die Net Name> <Other Die
Bump Name> <Current Die Name>(<Current Die Module Name>) <Current Die Net Name> <Current
Die Bump Name>
SIGNAL {
        top1(top1) A[0] A[0] top0(top0) A[0] A[0]
        top1(top1) A[10] A[10] top0(top0) A[10] A[10]
        top1(top1) A[11] A[11] top0(top0) A[11] A[11]
        ...
}
POWER {
        ...
}

GROUND {
        ...
}
```

Related Information

- read_bump_locations

write_spef_between_tsv_dies

```
write_spef_between_tsv_dies
out_file
```

Creates a top-level spef that describes the connectivity between dies. The RC is `0` in this SPEF file, but only inter-die connectivity is written. This spef file, together with the spef for each die, is the input of Tempus for static timing analysis.

This command is used after reading the stacked die config file (XML) using `read_tsv_config` as the die configuration is required to detect the connectivity between dies.

Parameters

out_file	Specifies the file name. *Data_type*: string, optional

Examples

- The following command creates a `0` load top-level spef file:

  ```
  write_spef_between_tsv_dies top.spef
  ```

Related Information

- `read_tsv_config`

write_design_stack_config

```
write_design_stack_config
[-help]
{[-xml StackedDieFileName] [-power_config_file PowerConnectionFile]}
```

Saves the xml file and power config file in the specified path.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `write_design_stack_config` parameter. For a detailed description of the command and all of its parameters, use the man command: `man write_design_stack_config`
`-power_config_file` *`PowerConnectionFile`*	Specifies the file name of the power connection.
`-xml` *`StackedDieFileName`*	Specifies the file name of the stacked die.

Related Information

- `read_design_stack_config`

read_design_stack_config

```
read_design_stack_config
[-help]
[-avoid_floorplan_change]
[-chip_name NameOfChip]
[-manufacturing_grid grid]
[-power_config_file PowerConnectionFile]
[-top_netlist VerilogFileName]
[-xml StackedDieFileName]
```

Loads the die stacking config file (*.xml) for Innovus to understand the 3D-IC top-level structure.

You can use this command before or after the `init_design` command. When you use `read_design_stack_config` before `init_design`, in addition to loading the die stacking information into the memory, it patches up the virtual techLEF of the 3DIC top cell and the Verilog netlist if the `init_lef_file` attribute is not set before `init_design`. If you use `read_design_stack_config` after `init_design`, when the top design (Netlist, tech LEF) has already been loaded into Innovus, but without XML. Without the XML, Innovus does not know that the top cell is the top of a 3DIC design. Therefore, XML is required to let Innovus know the vertical position and face up/down information of the dies for the GUI display and the contact layer mapping relationship.

When the top-level netlist is missing, `read_design_stack_config` can help generate a virtual netlist, but the pins are floating so that you can still load the 3D-IC design into Innovus with iHDB. Only top is in the virtual netlist. `read_design_stack_config` internally calls addInst to add other dies.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `read_design_stack_config` parameter. For a detailed description of the command and all of its parameters, use the man command: `man rread_design_stack_config`
`-avoid_floorplan_change`	When you specify this option, the floorplan and die instances will not be created.
`-chip_name` *`NameOfChip`*	Specifies the name to be used when the current die is instantiated.
`-manufacturing_grid` *`grid`*	Specifies the manufacturing grid.
`-power_config_file` *`PowerConnectionFile`*	Specifies the file name of the power connection.
`-top_netlist` *`VerilogFileName`*	Specifies the file name of the top-level netlist verilog.
`-xml` *`StackedDieFileName`*	Specifies the file name of the stacked die.

Related Information

- `init_design`
- `init_lef_files attribute`
- `write_design_stack_config`

44

Unix Commands

- flowtool
- innovus
- lef2oa
- nagelfar
- oa2lef

flowtool

```
  flowtool
[-db "type location [name] [view]"]
[-dir run_dir]
[-display {all | trunk}]
[-dist path_to_script]
[-enabled list_of_features]
[-files list_of_files]
[-from flow_or_flow_step]
[-inject_tcl tcl_code]
[-isolate]
[-jobs integer]
[-predict {summary | verbose | tcl | script}]
[-ref_run path_to_refernce_run]
[-reset]
[-run_tag name_of_run]
[-status]
[-to flow_or_flow_step]
```

Provides a way to run flows from the command line. In addition to providing another interface to existing Flowkit functionality, it enables multi-session, multi-tool and branching flows.

Note: TCL 8.5 or higher is required to run flowtool.

Parameters

`-db "type location [name] [view]"`	Specifies the database to be used to start the run. Flowtool distinguishes between the multiple DB types (such as OA and script) and tools (such as Innovus, Tempus, and Genus). This information is stored in four ways to determine how the DB will be read. The four values of the `-db` parameter specify those four ways of reading the DB. The `type` value refers to the supported DB types, which are mentioned below:
	• **TCL**
	Expanded form:
	`{tcl path_to_file null null}`
	Simplified form:
	`{file_that_ends_with.tcl}`
	• **ENC**
	Expanded form:
	`{enc path_to_enc.dat_dir null null}`
	Simplified form:
	`{path_to_dir_that_ends_with.dat}`
	• **RC**
	Expanded form:
	`{rc path_to_genus_db null null}`
	Simplified form:
	`{file_that_ends_with.db}`
	• **OA**
	Expanded form:
	`{oa path_to_cds.lib_location design view}`
	Simplified form:
	`{NONE}` since all four fields are required.

`-dir run_dir`	Changes to the named directory before doing anything else (including looking for starting scripts).
`-display {all | trunk}`	Filters the output to various types of flows.
`-dist path_to_script`	Specifies the distribution script and arguments to be pretended to the command line when executing tools. Type: `string`
`-enabled list_of_features`	Enables the specified features. Type: `string`
-files *list_of_files*	Specifies one or more files that describe the flow and flow steps. These files can be in the TCL or YAML syntax. When this parameter is used, `flowtool` searches for the default directory and master flow file name used for each flow type. *Default:* • `./scripts/run_flow.tcl`, when the TCL-based flow scripts are used. • `./scripts/flow.yaml`, when the YAML-based flow scripts are used.
`-from flow_or_flow_step`	Specifies the `flow` or `flow_step` name (inclusive) to start running the flow accepting a full dotted, comma-separated, and case-insensitive canonical path to a flow step object. Type: `string`
`-inject_tcl tcl_code`	Injects the specified TCL into every step. Type: `string`
`-isolate`	Runs each step in its own tool session.
`-jobs integer`	Specifies the maximum number of tasks to run in parallel. If this parameter is not specified, there is no limit to the number of flows, which can run simultaneously.
`-predict {summary | verbose | tcl | script}`	Specifies the prediction of the flow that will run (before executing the flow).

`-ref_run` *`path_to_refernce_run`*	If `-from` is specified, the `-ref_run` option points to a working directory/database of a previous run. Type: `string`
`-reset`	While trying to run a completed flow, this parameter starts it again from the beginning.
`-run_tag` *`name_of_run`*	Sets the `-dir` option to the same as the *`run_tag_value`*, if the `-dir` option is not specified. Additionally, it updates the tool startup code to include: `set_db flow_run_tag <value>` Type: `string`
`-status`	Prints the execution status (completed steps) of a run.
`-to` *`flow_or_flow_step`*	Specifies the `flow` or `flow_step` name (inclusive) to stop running the flow accepting a full dotted, comma-separated, and case-insensitive canonical path to a flow step object. Type: `string`

Related Information

Flowkit chapter of the *Innovus Stylus Common UI Migration Guide* document

innovus

```
innovus
[-abort_on_error]
[-batch]
[-cds_lib_file file.defs]
[-cpus value]
[-db dir | -ihdb dir]
[-disable_user_startup]
[-execute list_of_Tcl_commands]
[-files in_file_list]
[-help]
[-lic_multi_cpu lic_list]
[-lic_options lic_list]
[-lic_startup lic_list]
[-lic_startup_options lic_list]
[-lic_stack {1 | 2}]
[-log prefix]
[-no_cmd]
[-no_gui]
[-no_logv]
[-overwrite]
[-stylus]
[-version]
[-wait minutes]
```

Launches an Innovus™ Implementation System session. The `innovus` command starts one of the following products:

- Innovus Implementation System
- Innovus Implementation System Basic
- Virtuoso® Digital Implementation
- Virtuoso® Digital Implementation XL
- First Encounter® L
- First Encounter® XL

For an overview of the products and product licensing, see Product and Licensing Information chapter in the *Innovus Stylus Common UI User Guide.*

To start an Innovus session, type the `innovus` command with the appropriate parameters on the UNIX/Linux command line. If you type the command without parameters, the software starts in GUI mode and creates a log file and a command file. The system attempts to check out the license with the most functionality, then the license with the next most functionality, and so on.

Parameters

`-help`	Writes out the usage and options with brief descriptions.
`-abort_on_error`	Exits if an error is detected at any point. If any Tcl command returns `TCL_ERROR` to the top level, or any `Error (XXX): ...` message occurs, the process will exit.
`-batch`	Runs without interactive prompt. After all the options, including `-execute` and `-files`, are processed, the job exits.
`-cds_lib_file in_file`	
	Loads the specified file instead of the default `cds.lib` file in the current directory to obtain the names and locations of OpenAccess libraries. The `cds.lib` file is used to list the OpenAccess library names, and their physical locations.
`-cpus value`	Specifies the number of CPUs to be used for multi-threading on current host. This calls the appropriate Tcl command to set the local cpu count. This happens before any `-execute` or `-files` commands are processed. This may require extra multi-CPU licenses to be checked out (see `-lic_multi_cpu`).

`-db dir`	Specifies the DB directory to be read. This option calls `read_db` internally. Example: `innovus -db path/dbName.enc.dat` The specified DB directory is read before `-execute` or `-files` is invoked. **Note**: The database being restored with `-db` should not be an Integrated Hierarchical Database (iHDB). If you want to restore an iHDB database, use `-ihdb dir` instead. The `-db` and `-ihdb` options cannot be used together.
`-disable_user_startup`	Reads only the init file in the installation hierarchy. Does not read init files in the home or current directory. This option suppresses the default loading from the home and current directory, which can be used to guarantee there are no local over-rides to various project settings. See the "*Initialization Files*" section for the init files that are loaded by default.
`-execute list_of_Tcl_commands`	
	Specifies the Tcl commands to be executed at startup, as a quoted string.
`-files in_file_list`	Specifies one or more Tcl files to be sourced at startup, after `-execute` commands are processed. Set the shell envar `CDS_STYLUS_SOURCE_VERBOSE` to `0` if you don't want the `source` command output to default to verbose.
`-lic_multi_cpu lic_list`	
	Controls which licenses are to be used for higher multi-CPU usage in priority order from left to right. The allowed license names are: `invs_cpu`: Innovus CPU Accelerator Option `invs` - Innovus Implementation System `invsb` - Innovus Implementation System Basic `fel`: First Encounter® L `fexl`: First Encounter® XL For example, `-lic_multi_cpu "fel fexl"` will try to check out `fel` first, and then `fexl`. The licenses will not be checked out until a Tcl command tries to use extra cpus that require more licenses.
`-lic_options lic_list`	

	Specifies the order for dynamic checkout of extra option licenses when needed by a specific command. In some cases, more than one option license can be used to enable a given feature. You can control which options are allowed to be checked out, and in what priority (from left to right) when required during the run. The allowed license names are: `invs_hier` – Innovus Hierarchical Design Option `invs_ms` - Innovus Mixed Signal Option `invs_3d_ic` - Innovus 3D-IC Option `invs_3nm` - Innovus 3nm Option `invs_5nm` - Innovus 5nm Option `invs_7nm` - Innovus 7nm Option `invs_20nm` - Innovus 20nm Option `invs_10nm` - Innovus 10nm Option `invs_hfr` - Innovus HFR Option `invs_gigaplace` - Innovus GigaPlace XL Option `invs_gigaplace_gxl` - Innovus GigaPlace GXL Option `qrcaa` - Quantus Advanced Analysis GXL Option `enclp` - Encounter Low Power GXL Option `encan` - Encounter Advanced Node GXL Option `enccco` - Encounter Clock Concurrent Opt GXL Option `encng` - Encounter Next Generation (This license is used to enable beta features.) `vdixl_capacity` - VDI-XL Block Capacity Option `invs_pi` - Innovus Power Integrity Option `invs_ml` - Innovus Machine Learning Implementation Option `invs_ehfs` - Innovus Early Hierarchical Floorplan Synthesis Option `invs_dfm` - Innovus DFM Option `invs_sc` - Innovus Superconducting Option `invs_automotive` - Innovus Automotive Option `tpsxl` - Tempus Timing Signoff Solution XL `tpsl` - Tempus Timing Signoff Solution L `vtsxl` - Voltus IC Power Integrity Solution XL `vtsl` - Voltus IC Power Integrity Solution L `vtsxm` - Voltus IC Power Integrity Solution - XM `vtsesd` - Voltus IC Power Integrity Solution - ESD `vtsaa` - Voltus IC Power Integrity Solution Advanced Analysis GXL Option Option licenses can be checked out dynamically when a specific Tcl command requires it. This may cause the Tcl script to stall and wait for a license, or to exit if the wait period expires. For example, `-lic_options "encan encng enccco"` will only allow those three options to be checked-out dynamically, and in that priority order.
`-lic_startup` *`lic_list`*	

	Requests one of these licenses at startup in priority order from left to right. The supported licenses are: `invs` - Innovus Implementation System `invsb` - Innovus Implementation System Basic `fexl` - First Encounter® XL `vdixl` - Virtuoso® Digital Implementation XL. `vdi` - Virtuoso® Digital Implementation `fel`: First Encounter® L For example, `-lic_startup "fexl fel"` will try to check out `fexl` first, and then `fel`. The checkouts will be attempted repeatedly until one succeeds or the `-wait` period expires.
`-lic_startup_options` *`lic_list`*	

	Specifies additional option licenses needed at startup. If not provided, they will be checked out when needed during the run. The allowed license names are: `invs_hier` – Innovus Hierarchical Design Option `invs_ms` - Innovus Mixed Signal Option `invs_3d_ic` - Innovus 3D-IC Option `invs_3nm` - Innovus 3nm Option `invs_5nm` - Innovus 5nm Option `invs_7nm` - Innovus 7nm Option `invs_20nm` - Innovus 20nm Option `invs_10nm` - Innovus 10nm Option `invs_hfr` - Innovus HFR Option `invs_gigaplace` - Innovus GigaPlace XL Option `invs_gigaplace_gxl` - Innovus GigaPlace GXL Option `qrcaa` - Quantus Advanced Analysis GXL Option `enclp` - Encounter Low Power GXL Option `encan` - Encounter Advanced Node GXL Option `enccco` - Encounter Clock Concurrent Opt GXL Option `encng` - Encounter Next Generation (This license is used to enable beta features.) `vdixl_capacity` - VDI-XL Block Capacity Option `invs_pi` - Innovus Power Integrity Option `invs_ml` - Innovus Machine Learning Implementation Option `invs_ehfs` - Innovus Early Hierarchical Floorplan Synthesis Option `invs_dfm` - Innovus DFM Option `invs_sc` - Innovus Superconducting Option `invs_automotive` - Innovus Automotive Option `tpsxl` - Tempus Timing Signoff Solution XL `tpsl` - Tempus Timing Signoff Solution L `vtsxl` - Voltus IC Power Integrity Solution XL `vtsl` - Voltus IC Power Integrity Solution L `vtsxm` - Voltus IC Power Integrity Solution - XM `vtsesd` - Voltus IC Power Integrity Solution - ESD `vtsaa` - Voltus IC Power Integrity Solution Advanced Analysis GXL Option For example, `-lic_startup_options "enccco encan"` will try to check out both `enccco and encan` when Innovus starts up. The checkouts will be attempted repeatedly until it succeeds or the `-wait` period expires.
`-lic_stack {1 | 2}`	Specifies the number of licenses to stack for stackable licenses. Some startup licenses that have limited instance counts can be stacked so that multiple licenses can be checked-out to increase the instance limit. Innovus allows stacking of two `vdi` or `vdixl` licenses to increase the instance limit from 50,000 to 100,000. For example, `-lic_stack 2` combined with `-lic_startup vdi` would check out two vdi licenses to allow 100,000 instances. *Default*: `1`

`-log prefix`	Specifies the prefix for the `.log`, `.cmd`, and `.logv` files. If three arguments are given, such as `-log "a b c"`, then the three names are used exactly without any extension added. So, `-log "mylog mycmd mylogv"` will result in `mylog mycmd mylogv` files. The default for *prefix* is `innovus`. So if `-log` is not specified, `innovus.log`, `innovus.cmd`, and `innovus.logv` files will be created. If you specify `-log test`, the tool creates `test.log`, `test.cmd` and `test.logv` files. The `.log` file contains the normal logging output, the `.cmd` file contains the Tcl commands that were executed, and the `.logv` file contains a super-set of the .log output plus "verbose" debugging output that is needed only for detailed debugging. If the `.log` file already exists, an integer is added to the end of all the files to make them unique. This means if `-log test` is specified, the tool will create `test.log test.cmd test.logv` for the first run, `test.log1 test.cmd1 test.logv1` for the next run, and so on. Only the existence of .log is checked, the existence of the .cmd or .logv files is not checked. The `-overwrite` option can be used to disable this behavior and overwrite an existing .log file. If the *prefix* value has a period in it, the last extension is stripped off for the `.cmd` and `.logv` usage. For example, `-log out.log` will result in `out.log out.cmd out.logv` files, and `-log out.a.log` will result in `out.a.log out.a.cmd out.a.logv` files. However, if the last extension in the *prefix* value is `.gz`, it will be retained. For example, if you specify `-log test.gz`, the tool will create `test.log.gz`, `test.cmd.gz`, and `test.logv.gz` files. If you specify `-log "test.log.gz test.cmd test.logv.gz"`, the tool will not gzip all files but will create files exactly as specified: `test.log.gz`, `test.cmd`, and `test.logv.gz`. If `test.log` or `test.log.gz` already exists, an integer is added to make the new files unique: `test.log1.gz`, `test.cmd1`, and `test.logv1.gz`. If there is an additional extension before `.gz` , it is stripped off. You can disable any given file by using `/dev/null`. For example, `-log "my.log /dev/null /dev/null"` will only create `my.log`. *Default*: `innovus.log` **Note**: If you provide the log file name under a directory path and the directory path does not exist, Innovus will create the directory path and save the log file under the directory.
`-no_cmd`	Specifies that the `.cmd` file should not be created.
`-no_gui`	Prevents the GUI and Tk from initializing at startup so only the user type-in window, such as xterm, is visible. You can start the GUI later in the session with the command `gui_show`.
`-no_logv`	Specifies that verbose log file should not be created.
`-overwrite`	Overwrites an existing .log file (and associated `.cmd` and `.logv` files) instead of adding an integer to make the files unique.
`-ihdb dir`	Specifies that the Integrated Hierarchical Database (iHDB) directory is to be read, such as `xxx/my_design/placed`. This option calls `read_module_model` internally. Example: `innovus -ihdb iHDB_path/cell/tag` The specified iHDB directory is read before `-execute` or `-files` is invoked. The specified iHDB should already have been created with `write_module_model`. To learn more about the iHDB flow, refer to the "*Integrated Hierarchical Database*" section in the Hierarchical and Prototyping Flow chapter of the *Innovus Stylus Common UI User Guide* **Note**: If you want to restore a non-iHDB database, use the `-db dir` option instead. The `-db` and `-ihdb` options cannot be used together.
`-stylus`	Launches the Innovus Stylus Common UI interface. This changes the Tcl commands to the Stylus Common UI interface described in the Innovus Stylus Common UI Text Reference Manual and the Stylus Common UI Database Object Information manuals. See the Innovus Stylus Common UI Migration Guide for a summary of the changes.
`-version`	Returns the program version information.
`-wait minutes`	Specifies the number of minutes the system waits for a license to become available before exiting. *Default:* 0 (no wait time) *Value range:* 0 to 10,000

Initialization Files

By default, various initialization files are loaded at startup, if they exist. They can be used to configure the GUI, load utility Tcl files, or configure Innovus settings. The initialization files are read in the following sequence:

1. `~/.cadence/innovus/{gui.pref.tcl, gui.color.tcl, workspaces}` in the home directory if the GUI is enabled. These files are intended to be created by *View -> Set Preference* and *View -> All Colors -> Save* options for a specific user.

2. `.cadence/innovus/{gui.pref.tcl, gui.color.tcl, workspaces/}` in the current directory if the GUI is enabled. These files are intended to be created by *View -> Set Preference* and V*iew -> All Colors -> Save* options for any user working in this directory.
3. `<install_dir>/etc/innovus/enc.tcl` in the installation directory. This is intended for CAD teams that want to set Tcl parameters, customize the GUI, or add user-defined Tcl commands that affect all users of this release.
4. `~/enc.tcl` in the home directory. This is intended for user-specific setting that should affect only one user.
5. `./enc.tcl` in the current directory. This is intended for design-specific setting that should affect only designs in this directory.

Note: The GUI files are not loaded in the non-GUI mode.

If the `-stylus` parameter is specified, the initialization files are loaded in the following order instead:

1. `~/.cadence/innovus/{gui.pref.tcl, gui.color.tcl, workspaces}` in the home directory if the GUI is enabled
2. `.cadence/innovus/{gui.pref.tcl, gui.color.tcl, workspaces/}` in the current directory if the GUI is enabled.
3. `<install_dir>/etc/innovus/innovus.tcl` in the installation directory.
4. `~/.cadence/innovus/innovus.tcl` in the home directory.
5. `./.cadence/innovus/innovus.tcl` in the current directory.
6. `./innovus_startup.tcl` in the current directory.

Note: The `-stylus` usage sets the `source_verbose` root attribute to `true` at startup. The GUI preference init files (1 and 2 above) ignore this setting, but the non-GUI preference initialization files above (3, 4, 5, and 6) will use verbose logging by default. If the shell envar `CDS_STYLUS_SOURCE_VERBOSE` exists and has a boolean value, then the `source_verbose` attribute will be set to the envar value at startup. So you can turn off verbose logging at startup by using the following command:

```
setenv CDS_STYLUS_SOURCE_VERBOSE 0
```

Setting Up the Run-Time Environment

If *`install_dir`* is the location of your Innovus installation, you should set up your run-time environment like this:

- Add the *`install_dir`*`/bin` directory to your path. The `bin` directory has links to all the public executables in the install hierarchy.
- If you want the Innovus Stylus Common UI man pages to be available from the Unix `man` command, you can add *`install_dir`*`/share/innovus/stylus/man` to your `MANPATH` envar.
- If you want the Tcl man pages to be available from the Unix `man` command, you can add *`install_dir`*`/share/tcltools/man` to your `MANPATH` envar.

For example, you might add this to your startup shell script:

```
set install_dir = /tools/innovus16.2/lnx86

set path = ($install_dir/bin $path)

setenv MANPATH $install_dir/share/innovus/stylus/man:$install_dir/share/tcltools/man:$MANPATH
```

Note: When Innovus launches, it automatically adds either the legacy or the Innovus Stylus Common UI man pages (depending on the `-stylus` option) and the Tcl man pages to the beginning of the current `MANPATH` inside Innovus. Therefore, from within Innovus, the `man` command will see both sets of man pages before any other man pages.

Temporary File Locations

Each Innovus session creates its own temporary directory to store temporary files at the beginning of the run.

If you have the write permission, the tool creates `tmp_dir` under the current directory (.). If not, `tmp_dir` is created in `/tmp`. If the Unix envar `TMPDIR` is set, then the `tmp_dir` directory is created inside `$TMPDIR`.

The name of the `tmp_dir` directory will look like:

```
innovus_temp_[pid]_[hostname]_[user]_xxxxxx
```

Where `_xxxxxx` is a string added to make the directory unique. For example:

```
innovus_temp_10233_farm254_bob_nfp9ez
```

The temporary directory is automatically removed on exit or if the run terminated with a catchable signal (e.g. `SIGSEGV`).

Related Information

- Getting Started chapter of the *Innovus Stylus Common UI User Guide*
- Product and Licensing Information chapter of the *Innovus Stylus Common UI User Guide*
- Hierarchical and Prototyping Flow chapter in the *Innovus Stylus Common UI User Guide*
  - Stylus Hierarchical Database Flow
- Stylus Common UI Database Object Information
- Innovus Stylus Common UI Migration Guide

lef2oa

```
lef2oa
-lef file
-lib library
[-commentChar char]
[-compress]
[-compressLevel zLevel]
[-createFixedViaDefs]
[-dataModel version]
[-defaultRuleName name]
[-DMAttributes attributeList]
[-DMSystem sysName]
[-h | -help]
[-layerMap filelist]
[-layoutView viewName]
[-libDefFile fileList]
[-libPath path]
[-lockColors]
[-logFile file]
[-mapConflicts]
[-noInfo msgIds]
[-noWarning msgIds]
[-overwrite]
[-pinLabels]
[-pnrLibDataOnly]
[-shared]
[-suffix string]
[-techDMAttributes attrList]
[-techDMSystem sysName]
[-techLib library]
[-techLibPath path]
[-techRefs parentTechs]
[-templateFile file]
[-textHeight value]
[-textLayer value]
[-useFoundryInnovus]
[-view viewName]
[-v]
[-version]
```

The LEF to OpenAccess (`lef2oa`) translator can be used to convert LEF files into equivalent OpenAccess data. It supports three main styles of use:

1. Converting the technology information, such as layers, DRC rules, via rules, vias, and so on, into an OA tech file. This OA techfile is sufficient for Innovus use but is not really usable for Virtuoso because it does not contain information about custom layers below `metal1`, pcells, and so on. This techfile can be read by `techLoadDump` to create an ASCII techfile and then users can cut-and-paste the necessary information, such as DRC rules, into a Virtuoso OA PDK ASCII techfile to create a merged OA techfile.
2. Creating an incremental techfile using the `-pnrLibDataOnly` option that only has technology data that is dependent on the standard cell library, such as default routing rule (`LEFDefaultRouteSpec`), non-default rules (NDRs), sites, and so on.
3. Converting the LEF macros into OA abstract cellviews.

Innovus uses a Cadence version of the LEF to OpenAccess translator that does more than the public domain `lef2oa` executable. The main difference is that many DRC rules that are expressed as properties that begin with `LEF58_` (for example, `PROPERTY LEF58_SPACING "..." ;`) are understood and mapped directly to the correct OA foundry constraints. In general, the rules needed for 28nm nodes and older are supported, while newer rules for 20nm and smaller are not supported. If you have any unsupported LEF property rules, they can be translated using Innovus commands instead. To do so, read the LEF into Innovus and use the `write_oa_techfile` Innovus command.

Parameters

`-commentChar char`	Identifies the comment character. *Default*: `#`
`-compress`	Allows libraries to be compressed.

`-compressLevel` *zLevel*	Defines the compression level to be used. *Default*: 1
`-createFixedViaDefs`	Allows the creation of fixed via definitions.
`-dataModel` *version*	
	Specifies a data model value of 0, 1, 2, or 3, which determines the data model for the OpenAccess database and determines how option is omitted, the default data model value of 4 is used unless the OA_DATAMODEL environment variable is set or a file nam directory of your OpenAccess installation hierarchy. *Default*: `4`
`-defaultRuleName` *name*	
	Specifies the name of the constraint group (default: `LEFDefaultRouteSpec`) to generate in the technology database. If a constraint exists in a technology database being modified, the translation fails unless the `-overwrite` option is used. *Default*: `LEFDefaultRouteSpec`
`-DMAttributes` *attributeList*	
	Specifies a list of design management (DM) attributes that are associated with every library that this application creates. Attributes be no space on either side of the equal sign, two or more pairs must be space-separated, and the list must be enclosed in quotes.
`-DMSystem` *sysName*	Specifies the DM system to be used for the library, which can be any DM plug-in available to the user. The DM plug-ins included The default DM system is `oaDMSystem`, which automatically selects between `oaDMTurbo` or `oaDMFileSys`. Use of the `oaDMTurbo` plug
`-h | -help`	Displays usage information.
`-layerMap` *filelist*	Specifies a quoted, space-separated list of one or more layer mapping files to be used during translation. You must specify the co
`-layoutView` *viewName*	Specifies the view name used for FOREIGN container views. *Default:* `layout`
`-lef` *file*	Specifies the LEF file to import into the OpenAccess database.
`-lib` *library*	Specifies the name of the destination library. If the library exists, the library must contain a technology database or refer to a librar library exists, new designs are created for macro definitions. If you redefine a macro in the file, a new design is created by overwri exist, a library is created in the current working directory and will contain a technology database.
`-libDefFile` *fileList*	
	Specifies the library definitions (`lib.defs`) file to load. The path to the `lib.defs` file can be a full or relative path. If this option is on system is used to find and load the `lib.defs` file.
`-libPath` *path*	Specifies the path to the output library.
`-logFile` *file*	Specifies the log filename. If this option is omitted, the log filename defaults to `lef2oaCDS.log`.
`-lockColors`	Specifies that all shapes created from `PORT` and `PIN` definitions that have a `MASK` construct associated with them will have their lock color set.
`-mapConflicts`	Maps certain conflicts between incremental and base technologies to constraints and constraint parameter definitions.
`-noInfo` *msgIds*	Suppresses the specified INFO messages. *msgIds* is a quoted, space separated list of numbers. Each number in the list represer message you want to suppress. None of the numbers in the list may be zero. Suppressed messages do not appear on the termina
`-noWarning` *msgIds*	Suppresses the specified WARNING messages. *msgIds* is a quoted, space separated list of numbers. Each number in the list rep message you want to suppress. None of the numbers in the list may be zero. Suppressed messages do not appear on the termina total of WARNING messages displayed in the summary.
`-overwrite`	Replaces MACROS in an existing design with the translated data. If this option is omitted, the default behavior is to update the ex DBU value in an existing technology database.
`-pinLabels`	Creates pin labels in designs by using OpenAccess `oaAttrDisplay` objects on `oaTerms`. Labels are added on the `oaTerm` for each figure. See also the `-textLayer` and `-textHeight` options.

`-pnrLibDataOnly`	Maps only library-specific place and route information to an incremental tech, ignoring foundry rules. Recommended for convertin database that references a technology database that has complete foundry information. LEF constructs `DIRECTION` and `PITCH`, alo specified) are mapped to constraints in the `LEFDefaultRouteSpec` type constraint group. `VIAS` and `VIARULES` defined are added to t `NONDEFAULTRULES` are mapped to OA constraint groups, `SITES` are mapped to `oaSiteDefs`, and `MACROS` are mapped to abstract `oaDe`
`-shared`	Allows other applications to access the output library while this application is running.
`-suffix` *string*	Adds the specified suffix to the constraint group definition names.
`-techDMAttributes` *attrList*	Specifies a list of DM attributes to be associated with every technology library created.
`-techDMSystem` *sysName*	Specifies the DM system to be used for the technology library, which is either `oaDMTurbo` or `oaDMFileSys`. The default is `oaDMTurbo`
`-techLib` *library*	Specifies the name of a technology library to use when creating a new cell library. If the specified technology library does not exis information in the LEF file.
`-techLibPath` *path*	Specifies the path to be used for creating a technology library specified by the `-techLib` argument. The is only used when the libr exist.
`-techRefs` *libList*	Specifies the list of libraries that contain the master technology databases.
`-templateFile` *file*	Specifies a file containing arguments to `lef2oa`. You can specify a template file instead of entering a string of arguments on the co
`-textHeight` *value*	Specifies an integer value for the pin label text height in database units (DBU). The DBU value is set by `lef2oa` by using the DAT used, and this option is omitted, the pin label text height defaults to the DBU value in the technology database for maskLayout vie
`-textLayer` *value*	Specifies the layer used for pin label text. If `-pinLabels` is used, and this option is not specified, the pin label text layer defaults to technology database, it is added.
`-useFoundryInnovus`	Specifies that the LEF LAYER LEF58 properties constraints will be placed into the `foundry_innovus` constraint group. This optior tech files for use by the mixed signal flow. For more information on Rapid MSOA PDKs, refer to the Technology Data Preparation *Guide*.
`-view` *viewName*	Specifies the output view name. The default view name is `abstract`.
`-v`	Prints tool, format and library version information
`-version`	Prints tool and format version information

Related Information

- `oa2lef`
- `write_oa_techfile`
- Technology Data Preparation chapter in the *Mixed Signal (MS) Interoperability Guide*

nagelfar

```
nagelfar
[<nagelfar.tcl options>]
[file1.tcl file2.tcl …]
```

Does the “lint type” checking of Innovus Tcl script files.

It finds many common Tcl errors, such as misspelled commands, using a Tcl variable before setting it.

The check-syntax command is a wrapper around the public-domain Tcl linter named nagelfar. Nagelfar automatically loads the Innovus syntax and plugin files and is therefore aware of Innovus commands and options. Nagelfar is available from http://sourceforge.net/projects/nagelfar. A recent version of nagelfar is included in the installation hierarchy at: `<install_dir>/share/nagelfar`.

Nagelfar is a wrapper around `<install_dir>/share/nagelfar/nagelfar.tcl`

The wrapper automatically loads the correct Innovus nagelfar legacy syntax file and plugin file so that Innovus commands and options are understood by nagelfar. The files are loaded from:

```
<install_dir>/share/innovus/nagelfar/innovus.syntaxdb.tcl
<install_dir>/share/innovus/nagelfar/innovus.plugin.tcl
```

If you want to check the Stylus Common UI command syntax, you need use the nagelfar files from here:

```
<install_dir>/share/commonui/nagelfar/innovus.commonui.syntaxdb.tcl
<install_dir>/share/commonui/nagelfar/innovus.commonui.plugin.tcl
```

and call negalfar.tcl directly from the Linux prompt to avoid loading the legacy files like this:

```
set install_dir = <install_dir>

set nag_dir = $install_dir/share/nagelfar

set cui_dir = $install_dir/share/commonui/nagelfar

$nag_dir/nagelfar.tcl \

-s $cui_dir/innovus.commonui.syntaxdb.tcl \

-plugin $cui_dir/innovus.commonui.plugin.tcl \

<other nagelfar options> [file1.tcl file2.tcl …]
```

You can check the nagelfar documentation in the installation directory for details about its options, the checks it performs, and the steps to turn on or off various checks.

To run this command, you must have tclsh in your search path.

You can type `nagelfar -help` to see all the nagelfar.tcl options.

Options	Description
`-encoding <enc>`	Reads script with this encoding.
`-exitcode`	Returns status code 2 for any error or 1 for warning.
`-filter <p>`	Specifies to suppress any message that matches the global pattern.
`-glob <pattern>`	Adds matching files to scriptfiles to check.
`-gui`	Starts with GUI even when files are specified.
`-H`	Specifies a prefix each error line with file name.
`header <file>`	Creates a "header" file with syntax info for scriptfiles.
`-html`	Generates html-output.
`-instrument`	Instruments source file for code coverage.
`-markup`	Markup source file with code coverage result.
`-novar`	Disables variable checking.
`-prefix <pref>`	Specifies the prefix for line anchors (html output)
`-quiet`	Suppresses non-syntax output.

`-s <dbfile>`	Includes a database file. You can include more than one database file. If <dbfile> is "_", it adds the default database.
`-severity <level>`	Sets severity level filter to N/W/E (default N).
`-strictappend`	Enforces having an initialized variable in (l)append.
`-tab <size>`	Specifies the Tab size, default is 8.
`-WelseN`	Enforces else keyword. Default 1.
`-WexprN`	Sets expression warning level to N.
`-WsubN`	Sets subcommand warning level to N.
`1`	Does not warn on single commands. "if [apa] {...}" is ok.
`1 (def)`	Warns about shortened subcommands.
`2 (def)`	Warns about any unbraced expression.

Example

- The following example show the usage of `nagelfar` for file `a.tcl`:

```
linux> nagelfar -H a.tcl
Checking file a.tcl
a.tcl: 2: W Unknown command "typo_in_cmd_name"
a.tcl: 3: E Bad option -typo_in_opt_name to "routeDesign"
```

Related Information

- `check_syntax` is the corresponding Tcl command that can be used directly inside Innovus.

oa2lef

```
oa2lef
-lef file
-lib library
[-cell cells]
[-h | -help]
[-libDefFile filename]
[-lockedColorsOnly]
[-logFile file]
[-noInfo msgIds]
[-noTech]
[-noWarning msgIds]
[-templateFile file]
[-useFoundryInnovus]
[-views viewName]
[-v]
[-ver version]
[-version]
```

This translator can be used to convert OpenAccess data into equivalent LEF data.

Innovus uses a Cadence version of `oa2lef` that does more than the public domain `oa2lef` executable. The main difference is that many DRC rules that are expressed as properties that begin with `LEF58_` (for example, `PROPERTY LEF58_SPACING "..." ;`) are understood and mapped to the correct LEF syntax. In general, the rules needed for 28nm nodes and older are supported, while newer rules for 20nm and smaller are not supported. If you have any unsupported LEF property rules, they can be translated into LEF using Innovus commands instead. To do so, read the OA techfile into Innovus, and use the Innovus `write_lef_library` command.

Parameters

`-cell` *cells*	Specifies a space-separated list of cells to translate.
`-help or -h`	Displays usage information.
`-lef` *file*	Specifies the output LEF file name.
`-lib` *library*	Specifies the name of the source library, containing a technology database.
`-libDefFile` *filename*	Specifies the `lib.defs` file to load. The path to the `lib.defs` file can be a full or relative path. If this option is omitted, the default search mechanism of the DM system is used to find and load the `lib.defs` file.
`-lockedColorsOnly`	Prints the colors of only the shapes with a locked state of `true` when mapping the oaFigs or oaVias belonging to oaPins to the LEF PINs and PORTs.
`-logFile` *file*	Specifies the log filename. If this option is omitted, the log filename defaults to `oa2lef.log`.
`-noInfo` *msgIds*	Suppresses the specified INFO messages. msgIds is a quoted, space-separated list of numbers. Each number in the list represents the numerical portion of the ID for the message you want to suppress. None of the numbers in the list may be zero. Suppressed messages do not appear on the terminal or in the log file.
`-noTech`	Specifies that the technology information is not output.
`-noWarning` *msgIds*	Suppresses the specified WARNING messages. *msgIds* is a quoted, space-separated list of numbers. Each number in the list represents the numerical portion of the ID for the message you want to suppress. None of the numbers in the list may be zero. Suppressed messages do not appear on the terminal or in the log file and are not included in the total of WARNING messages displayed in the summary.
`-templateFile` *file*	Specifies a file containing arguments to `oa2lef`. You can specify a template file instead of entering a string of arguments on the command line.
`-useFoundryInnovus`	Prints the `foundry_innovus` constraint group as LEF LAYER LEF58 properties. These constraints are found only in Rapid MSOA tech files that are used for the Cadence mixed signal implementation solution. For more information on Rapid MSOA PDKs, refer to the Technology Data Preparation chapter in the *Mixed Signal (MS) Interoperability Guide*.
`-v`	Prints tool, format, and library version information.
`-ver` *version*	Specifies the LEF version to write, which can be 5.3, 5.4, 5.5, 5.6, 5.7 or 5.8. The default version is 5.7.
`-version`	Prints tool and format version information
`-views` *viewName*	Specifies a space-separated list of views to translate. Only the first view name for a particular cell is used; all others are ignored.

Related Information

- `lef2oa`
- `write_lef_library`
- Technology Data Preparation chapter in the *Mixed Signal (MS) Interoperability Guide*

45

Unified Safety Format Commands

Midas Unified Safety Format (USF) Physical is loaded at startup. To enable USF in Innovus, you must set the `safety_midas_enable` attribute to `physical`.

```
set_db safety_midas_enable physical
```

You will see a message like this:

```
Cadence Midas USF - Physical
Copyright 2022 Cadence Design Systems, Inc. All rights reserved worldwide.
Cadence and the Cadence logo are registered trademarks of Cadence Design Systems Inc.
in the United States and other countries.
Midas USF - Physical Version: 22.03.002
1 physical
```

ⓘ Once USF Physical is loaded, it is not possible to disable or reload it in the current session.

- USF File Commands
- USF Support Commands

USF File Commands

- apply_safety_mechanism
- create_esubpart
- create_failure_mode
- create_safety_mechanism
- set_safety_mechanism_rules_dcls
- set_safety_mechanism_rules_tmr

apply_safety_mechanism

```
apply_safety_mechanism
[-help]
safety_mechanism
[-generated]
-to failure_mode
```

Maps a safety mechanism to one or more failure modes.

This command should be applied only via a USF file that is read in using `read_usf`.

Parameters

`-help`	Prints a brief description of the `apply_safety_mechanism` command. For a detailed description of the command, use the `man` command: `man apply_safety_mechanism`.
safety_mechanism	Specifies the safety mechanism identifier. *Data_type*: string, required
`-generated`	Specifies that the additional logic required to create this safety mechanism has already been inserted into the design. *Data_type*: bool, optional
`-to` *failure_mode*	Specifies the failure mode identifier to which the specified safety mechanism should be associated. **Note**: One safety mechanism can be mapped to multiple failure modes. *Data_type*: string, required

Example

- The following command applies the `SM_TMR` safety mechanism to the `FM_22` failure mode:

  ```
  apply_safety_mechanism SM_TMR -to FM_22
  ```

Related Information

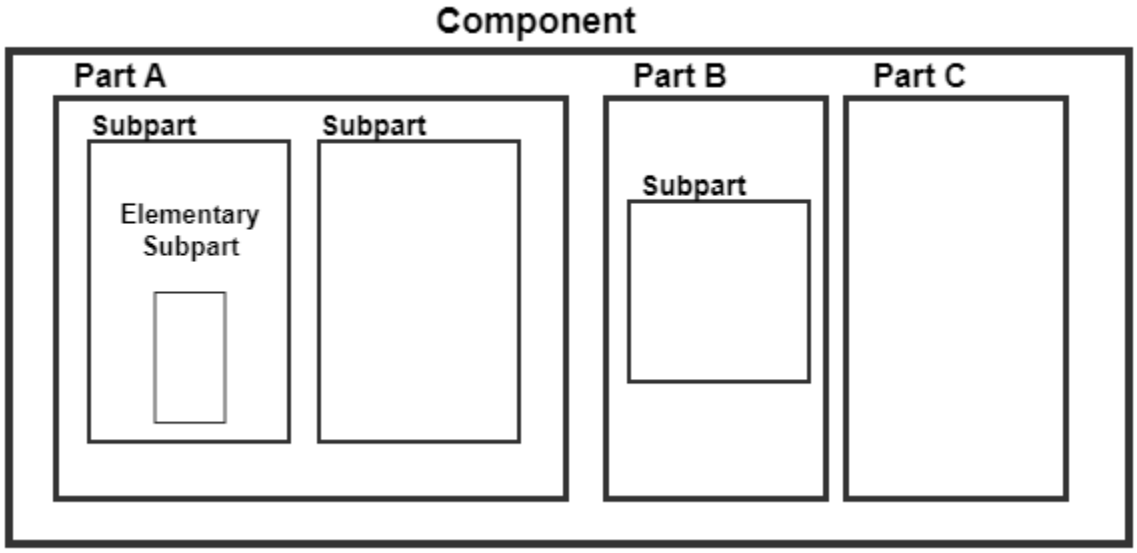
- check_safety_mechanism
- create_esubpart
- create_failure_mode
- create_safety_mechanism
- read_usf
- safety Category Attributes
- set_safety_mechanism_rules_dcls
- set_safety_mechanism_rules_tmr

create_esubpart

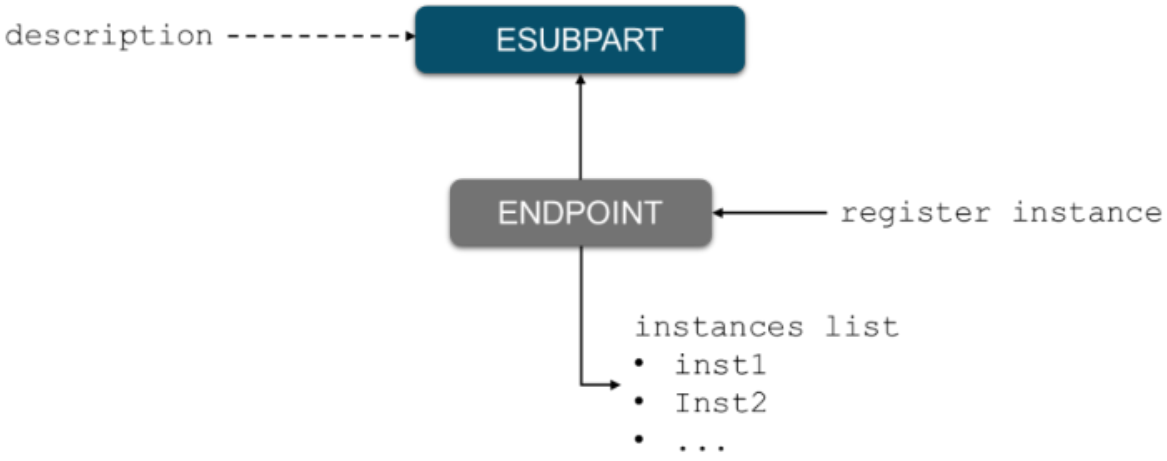
```
create_esubpart
[-help]
esp
[-description description]
-inst string
-register_endpoint string
```

Creates an elementary subpart, which is the smallest hardware subpart considered in safety analysis.

The diagram below represents a component with its parts, subparts, and elementary subparts.



An elementary subpart is a group of instances identified with a descriptor. The instances are associated to reference an endpoint. In the current implementation, the endpoint can only be a register.



The `create_esubpart` command should be applied only via a USF file that is read in using `read_usf`.

Parameters

`-help`	Prints a brief description of the `create_esubpart` command. For a detailed description of the command, use the `man` command: `man create_esubpart`.
esp	Specifies the identifier to be associated with the elementary subpart being created. *Data_type*: string, required
`-description` *description*	
	Specifies a textual description for the elementary subpart being created. *Data_type*: string, optional
`-inst` *string*	Specifies the instances to be mapped to the elementary subpart. *Data_type*: string, required
`-register_endpoint` *string*	
	Specifies a reference instance for the endpoint. *Data_type*: string, required

Example

- The following command creates an elementary subpart named with the specified description, endpoint, and instances:

```
create_esbupart ESP_VOT_0 -description "automatically generated elementary subpart"
 -register_endpoint { xsv_fpga_top/or1200_top/or1200_cpu/or1200_fpu/s_exp_diff_reg\[5\] }
 -inst { xsv_fpga_top/or1200_top/or1200_cpu/or1200_fpu/s_exp_diff_reg\[5\]_tv_an1
         xsv_fpga_top/or1200_top/or1200_cpu/or1200_fpu/s_exp_diff_reg\[5\]_tv_an2
         xsv_fpga_top/or1200_top/or1200_cpu/or1200_fpu/s_exp_diff_reg\[5\]_tv_or1
         xsv_fpga_top/or1200_top/or1200_cpu/or1200_fpu/s_exp_diff_reg\[5\]_tv_or2
 }
```

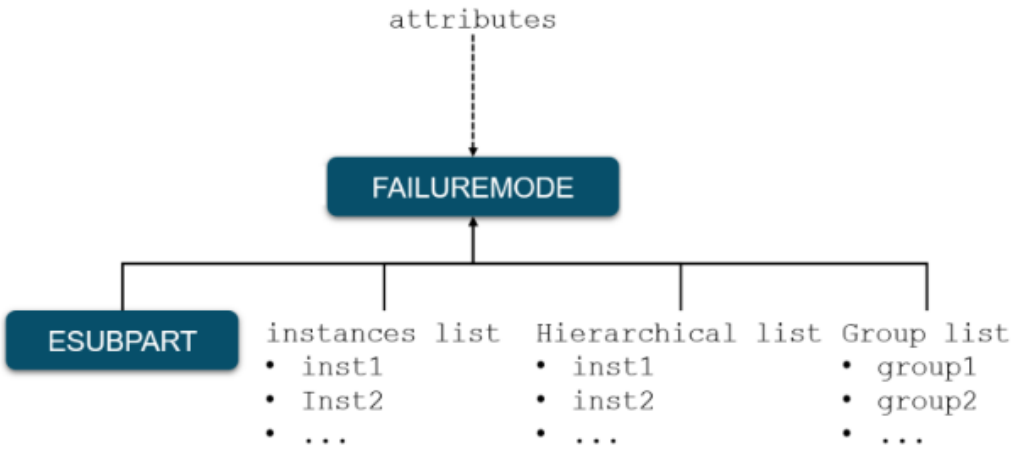
Related Information

- apply_safety_mechanism
- check_safety_mechanism
- create_failure_mode
- create_safety_mechanism
- read_usf
- safety Category Attributes
- set_safety_mechanism_rules_dcls
- set_safety_mechanism_rules_tmr

create_failure_mode

```
create_failure_mode
[-help]
failuremode
[-attributes string]
[-description description]
{-hinst string | -inst string | -esubpart string | -group string}
```

Defines a failure mode. The failure mode groups a given set of logic that is relevant from the safety point of view. For example, it could be the logic that needs to be protected by one or more safety mechanisms or the logic that is describing a safety mechanism that has already been implemented.



The `create_failure_mode` command should be applied only via a USF file that is read in using `read_usf`.

Parameters

`-help`	Prints a brief description of the `create_failure_mode` command. For a detailed description of the command, use the `man` command: `man create_failure_mode`.
`failuremode`	Specifies the identifier to be associated to the failure mode being created. *Data_type*: string, required
`-attributes` *`string`*	Specifies the optional attributes attached to the failure mode. *Data_type*: string, optional
`-description` *`description`*	Specifies a textual description for the failure mode being created. *Data_type*: string, optional
`-esubpart` *`string`*	Specifies a list of elementary subparts. *Data_type*: string, required
`-group` *`string`*	Specifies the groups to be mapped to the failure mode. *Data_type*: string, required
`-hinst` *`string`*	Specifies the hierarchical instance(s) to be mapped to the failure mode. *Data_type*: string, required
`-inst` *`string`*	Specifies the instances to be mapped to the failure mode. *Data_type*: string, required

Example

- The following command creates a failure mode named `FM_22` with the specified description and hierarchical instance:

```
create_failure_mode FM_22 -description "FPU Adder produces incorrect data due to
fault into the FPU prenorm addsubtracter"
     -hinst
xsv_fpga_top/or1200_top/or1200_cpu/or1200_fpu/fpu_arith/fpu_prenorm_addsub
```

Related Information

- apply_safety_mechanism
- check_safety_mechanism
- create_esubpart
- create_safety_mechanism
- read_usf
- safety Category Attributes
- set_safety_mechanism_rules_dcls
- set_safety_mechanism_rules_tmr

create_safety_mechanism

```
create_safety_mechanism
[-help]
safety_mechanism
[-class {aou hw sw aou_hw aou_sw}]
[-description description]
[-type {tmr dcls}]
```

Defines a new safety mechanism. A safety mechanism is extra hardware inserted into your design to bypass, detect, or correct faults.

This command should be applied only via a USF file that is read in using `read_usf`.

Parameters

`-help`	Prints a brief description of the `create_safety_mechanism` command. For a detailed description of the command, use the `man` command: `man create_safety_mechanism`.
`safety_mechanism`	Specifies a name for the safety mechanism being defined. *Data_type*: string, required
`-class {aou hw sw aou_hw aou_sw}`	Specifies the supported safety mechanism classes. The possible values are: • `aou`: assumption of use • `hw`: hardware • `sw`: software • `aou_hw`: assumption of use hardware • `aou_sw`: assumption of use software *Default*: `hw` *Data_type*: enum, optional
`-description` *`description`*	Specifies a description for the safety mechanism being defined. *Data_type*: string, optional
`-type {tmr dcls}`	Specifies the type of the safety mechanism being defined. The options are: • `tmr` - triple modular redundancy • `dcls` - dual-core lockstep *Default*: `tmr` *Data_type*: enum, optional

Example

- The following command creates a safety mechanism named, `SM_TMR`, of the specified class and type:

```
create_safety_mechanism SM_TMR -description "Triple Modular Redundant Flip Flops" -type tmr -class hw
```

Related Information

- `apply_safety_mechanism`
- `check_safety_mechanism`
- `create_esubpart`
- `create_failure_mode`
- `read_usf`
- `safety` Category Attributes
- `set_safety_mechanism_rules_dcls`
- `set_safety_mechanism_rules_tmr`

set_safety_mechanism_rules_dcls

```
set_safety_mechanism_rules_dcls
[-help]
[-exclusive_groups {{group_A1 {group_A2 group_A3 ...}} {group_B1 group_B2
group_B3 ...} ... }]
[-isolate_clock {* | group_A1 group_A2 ...}]
[-spacing d]
```

Defines the separation rules to be applied to the dual core lockstep (DCLS) groups in the design.

Each time the `set_safety_mechansim_rules_dcls` command is issued, the tool clears any previously set DCLS-associated attributes across all groups prior to setting them in the way the latest command dictates. Therefore, this command cannot be used multiple times to accumulate a set of rules.

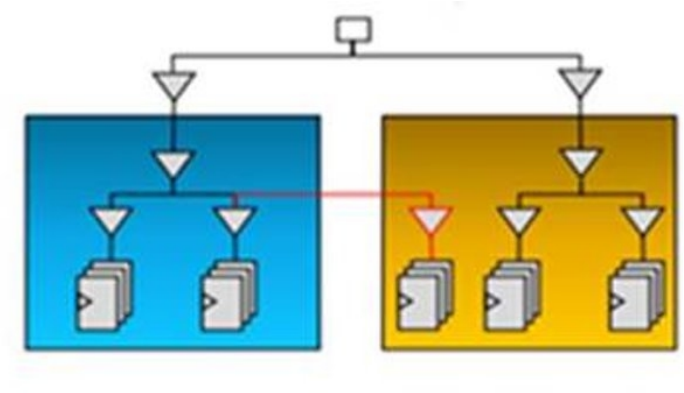
This `set_safety_mechansim_rules_dcls` command may be interactively applied at the command line. That is, it is not restricted to be read in as part of a USF file using `read_usf`. However, if the command is not in a USF file that is being read, it must be applied only after DCLS-related USF information has been read because it depends on this information to function correctly.

Attempting to use this command before a design is loaded or before the groups listed in arguments are defined will cause an error to be issued. In addition, any groups mentioned in this command must already be defined as DCLS groups using a combination of the `create_failure_mode`, `create_safety_mechanism` (with type `dcls`), and `apply_safety_mechanism` commands.

Parameters

`-help`	Prints a brief description of the `set_safety_mechanism_rules_dcls` command. For a detailed description of the command, use the `man` command: `man set_safety_mechanism_rules_dcls`.
`-exclusive_groups {{group_A1 {group_A2 group_A3 ...}} {group_B1 group_B2 group_B3 ...} ... }`	

	Specifies a list of mutually exclusive DCLS group lists. This parameter is mandatory if the `-spacing` parameter is being used. Wildcards are not allowed in the `-exclusive_groups` arguments. If a group is referenced more than once in the `-exclusive_groups` arguments, an error will be issued. In the argument string above, `{group_A1 {group_A2 group_A3 ...}` is a master/checker arrangement, where *group_A1* is the "master" group and *group_A2* and *group_A3* are the "checker" groups. This means that the checker groups must be separated from the master group by a distance of at least *d* as specified with the `-spacing` parameter. However, a checker group may overlap other checker groups. When a master/checker arrangement is specified, there can be only one master and there must be at least two checkers. In the `{group_B1 group_B2 group_B3 ...}` part of the argument string, *group_B1*, *group_B2*, and *group_B3* are all mutually exclusive. This means that they must all be separated from one another by a distance of at least *d* specified with the `-spacing` parameter. Each list of mutually exclusive groups must have at least two members or else an error will be issued.
`-isolate_clock {* | group_A1 group_A2 ...}`	

	Specifies a list of groups where clock isolation is required. This argument is optional. Clock isolation is specific to each listed group. If specified for a group, it isolates that group from the rest of the clock network in the design. Clock isolation is not just about separating two groups. For example, in the diagram below, the path shown in red will not occur for the groups with isolated clocking.  If * is specified, clock isolation is enacted for all the DCLS groups in the design.
`-spacing` *d*	Defines in microns the minimum spacing between exclusive DCLS groups. This parameter is mandatory if the `-exclusive_groups` parameter is used. The distance `d` here is expected to be a positive rational number. The specified distance is rounded to the nearest 0.001 value. If the distance specified is a negative number, a warning is issued and the setting is changed to `0`.

Examples

- The following command specifies that groups `A` and `B` are mutually exclusive to one another. The group boundaries must be at least 5 microns apart:
  ```
  set_safety_mechanism_rules_dcls -exclusive_groups { {A B} } -spacing 5
  ```
- The following command specifies that groups `A`, `B`, and `C` are mutually exclusive. These groups must be at least 10 microns apart:
  ```
  set_safety_mechanism_rules_dcls -exclusive_groups { {A B C} } -spacing 10
  ```
- The following command specifies that group `W` must be at least 5 microns from groups `X`, `Y`, and `Z`. However, `X`, `Y`, and `Z` may overlap one another:
  ```
  set_safety_mechanism_rules_dcls -exclusive_groups { {W {X Y Z}} } -spacing 5
  ```
- The following command specifies that groups `A`, `B`, and `C` must be at least 5 microns apart from one another and group `W` must be at least 5 microns from groups `X`, `Y`, and `Z`. However, `X`, `Y`, and `Z` may overlap one another. In addition, groups `A`, `B`,`C`, `W`, `X`, `Y`, and `Z` are to have isolated clocks:
  ```
  set_safety_mechanism_rules_dcls -exclusive_groups { {A B C} {W {X Y Z}} } -spacing 5
  -isolate_clock { * }
  ```

Related Information

- `apply_safety_mechanism`
- `check_safety_mechanism`
- `create_esubpart`
- `create_failure_mode`
- `create_safety_mechanism`
- `read_usf`
- `safety` Category Attributes
- `set_safety_mechanism_rules_tmr`

set_safety_mechanism_rules_tmr

```
set_safety_mechanism_rules_tmr
[-help]
{[-spacing double] [-spacing_x dx] [-spacing_y dy] [-isolate_clock {none unique_driver}] }
[-spacing double | -spacing_x dx]
[-spacing double | -spacing_y dy]
```

Defines the triple modular redundancy (TMR) safety-mechanism rules that need to be enforced during implementation.

The rules concern the physical spacing between TMR flip-flops and clock isolation of TMR flip-flops. The `set_safety_mechanism_rules_tmr` command may be applied interactively at the command line. That is, it is not restricted to be read in as part of a USF file using `read_usf`.

Each time the `set_safety_mechansim_rules_tmr` command is issued, the tool first clears any previously set TMR rules. This means that the `set_safety_mechansim_rules_tmr` command cannot be used multiple times to accumulate a set of rules.

One of the following parameters must be specified with the command:

- `-spacing`
- `-spacing_x` and/or `-spacing_y`
- `-isolate_clock`

Additionally, the `-spacing` parameter is mutually exclusive to the `-spacing_x` and `-spacing_y` parameters.

Parameters

`-help`	Prints a brief description of the `set_safety_mechanism_rules_tmr` command. For a detailed description of the command, use the `man` command: `man set_safety_mechanism_rules_tmr`.
`-isolate_clock {none unique_driver}`	

	Controls whether or not the three flops in a TMR group may share the same leaf clock buffer. Supported values are: • `none` - Specifies that the three flops in a TMR group may also share the same leaf clock buffer. • `unique_driver` - Specifies that the clock pin of every TMR flop should be directly driven by its own unique clock cell with fanout equal to `1`. *Default*: `none` *Data_type*: enum, required
`-spacing` *double*	Specifies the minimum radial spacing in µm between a group of TMR flip-flops sharing the same parent. This parameter is mutually exclusive to the `-spacing_x` and `-spacing_y` parameters. The spacing is expected to be a positive rational number. The specified value is rounded to the nearest 0.001. If a negative number is given, a warning is issued and the `spacing` value is set as `0`. *Data_type*: double, required
`-spacing_x` *dx*	Specifies the minimum spacing in the X dimension in µm between a group of TMR flip-flops sharing the same parent. This parameter is mutually exclusive to the `-spacing` parameter. The specified value is rounded to the nearest 0.001. If a negative number is given, a warning is issued and the `-spacing_x` value is set as `0`. *Data_type*: double, required
`-spacing_y` *dy*	Specifies the minimum spacing in the Y dimension in µm between a group of TMR flip-flops sharing the same parent. This parameter is mutually exclusive to the `-spacing` parameter. The specified value is rounded to the nearest 0.001. If a negative number is given, a warning is issued and the `-spacing_y` value is set as `0`. *Data_type*: double, required

Example

- The following command specifies a 5µm radial spacing between TMR flip-flops and requires isolation of clock pins.

```
set_safety_mechansim_rules_tmr -spacing 5 -isolate_clock unique_driver
```

Related Information

- apply_safety_mechanism
- check_safety_mechanism
- create_esubpart
- create_failure_mode
- create_safety_mechanism
- read_usf
- safety Category Attributes
- set_safety_mechanism_rules_dcls

USF Support Commands

- check_safety_mechanism
- read_safety_tmr
- read_usf
- reset_usf
- usf_version
- write_usf

check_safety_mechanism

```
check_safety_mechanism
[-help]
[-out_file file_name]
```

Checks all the safety mechanisms and prints a report, optionally writing it to the specified file as well.

The report contains a list of:

- All the safety mechanisms
- The safety mechanism violations found for each safety mechanism, grouped by type

See Report Content for details.

Parameters

`-help`	Prints a brief description of the `check_safety_mechanism` command. For a detailed description of the command, use the `man` command: `man check_safety_mechanism`.
`-out_file` *file_name*	Writes the report to the specified file. If omitted, no report file is generated. *Data_type*: string, optional

Report Content

The report contains one section per safety mechanism, listing all violations (if any) of that safety mechanism, grouped by type.

Safety mechanisms of type `dcls` (dual-core lockstep) have the following violation types:

- dcls group overlap violations: Two exclusive dcls groups overlap.
- dcls group spacing violations: Two exclusive dcls groups are closer than permitted.
- dcls inst placement violations: An inst belonging to a dcls group is placed outside of that group's boundary.
- dcls internal net routing violations: A net connected only to the insts inside a single dcls group is routed outside that group's boundary.
- dcls interface net routing violations: A net connected to insts both inside and outside a dcls group is routed inside the boundary of an exclusive group.
- dcls clock isolation violations: A clock net connected to a dcls group is driving loads outside that group.

Safety mechanisms of type `tmr` (triple modular redundancy) have the following violation types:

- tmr spacing violations: Two tmr insts from the same safety mechanism are placed closer than permitted.
- tmr clock isolation violations: A clock net driving a tmr inst is also driving other insts.

Example

- The following command checks all the safety mechanisms and writes the report to the `safety_report.rpt` file:

  ```
  check_safety_mechanism -out_file safety_report.rpt
  ```

Related Information

- create_safety_mechanism
- read_safety_tmr
- read_usf
- reset_usf
- safety Category Attributes
- usf_version
- write_usf

read_safety_tmr

```
read_safety_tmr
[-help]
tmr_info_file
-failure_mode failure_mode_name
-safety_mechanism safety_mechanism_name
```

Reads the triple modular redundancy (TMR) information file. For designs with TMR logic already inserted, either in the register-transfer level (RTL) or with third party tools, this command identifies the existing TMR logic and assimilates it into the Unified Safety Format (USF).

You can issue the `read_safety_tmr` command multiple times to build up information in cases where there are multiple different safety mechanism names or failure mode names. However, this may be done only for new safety mechanism and failure mode names. You cannot accumulate TMR information by reusing a safety mechanism or failure mode name that has been already read in with the `read_safety_tmr` command.

Parameters

`-help`	Prints a brief description of the `read_safety_tmr` command. For a detailed description of the command, use the `man` command: `man read_safety_tmr`.

tmr_info_file	Specifies the name of the TMR information file. This is a text file listing each flip-flop and its accompanying cloned flip-flops where TMR has already been applied. Optionally, the file may also list the names of the voter cells that are used.
	
	The TMR information file has the following format:
	
	`<parent_FF1> <clone_FF1A> <clone_FF1B> <voter1A> <voter1B>`
	`...`
	`<parent_FF2> <clone_FF2A> <clone_FF2B> <voter2A> <voter2B>`
	`...`
	`...`
	
	The instance names may be supplied with the full path including the design name at the root. If the design name is not given at the root of the instance path for a given instance, that instance is assumed to be in the current design. The instance names are separated by white space.
	
	Comments are allowed in this file and must be preceded by #.
	
	Blank lines are also allowed.
	
	Data_type: string, required

`-failure_mode` *failure_mode_name*	
	Specifies the name of the failure mode for parent flops. The failure mode for clone flops will be *<failure_mode_name>*`_clones`. The failure mode for voter cells will be *<failure_mode_name>*`_voters`. *Data_type*: string, required
`-safety_mechanism` *safety_mechanism_name*	
	Specifies the name of the safety mechanism for the TMR logic. *Data_type*: string, required

Related Information

- `check_safety_mechanism`
- `read_usf`
- `reset_usf`
- `safety` Category Attributes
- `usf_version`
- `write_usf`

read_usf

```
read_usf
[-help]
USF_file
```

Reads in a Unified Safety Format (USF) file specifying the safety intent of your design.

Parameters

`-help`	Prints a brief description of the `read_usf` command. For a detailed description of the command, use the `man` command: `man read_usf`.
`USF_file`	Specifies the name of the USF file to be read. *Data_type*: string, required

Example

- The following command reads in the USF file named `myFile.usf`:

  ```
  read_usf myFile.usf
  ```

Related Information

- `check_safety_mechanism`
- `read_safety_tmr`
- `reset_usf`
- `safety` Category Attributes
- `usf_version`
- `write_usf`

reset_usf

```
reset_usf
[-help]
```

Resets the Unified Safety Format (USF) Physical safety information and related database attributes completely.

Parameters

`-help`	Prints a brief description of the `reset_usf` command. For a detailed description of the command, use the `man` command: `man reset_usf`.

Related Information

- `check_safety_mechanism`
- `read_safety_tmr`
- `read_usf`
- `safety` Category Attributes
- `usf_version`
- `write_usf`

usf_version

```
usf_version
[-help]
```

Retrieves the version of the USF Physical being used to interpret the USF commands.

Parameters

`-help`	Prints a brief description of the `usf_version` command. For a detailed description of the command, use the `man` command: `man usf_version`.

Related Information

- `check_safety_mechanism`
- `read_safety_tmr`
- `read_usf`
- `reset_usf`
- `safety` Category Attributes
- `write_usf`

write_usf

```
write_usf
[-help]
USF_file
```

Writes the safety intent of your design to the specified Unified Safety Format (USF) file.

Parameters

`-help`	Prints a brief description of the `write_usf` command. For a detailed description of the command, use the `man` command: `man write_usf`.
`USF_file`	Specifies the name of the USF file to be saved. *Data_type*: string, required

Example

- The following command saves the USF information in the `myFile.usf` file.

  ```
  write_usf myFile.usf
  ```

Related Information

- `check_safety_mechanism`
- `read_safety_tmr`
- `read_usf`
- `reset_usf`
- `safety` Category Attributes
- `usf_version`

46

Wire Edit Commands and Attributes

- convert_special_routes_for_eco
- create_fillet
- delete_fillet
- delete_routes
- delete_secondary_pg_net
- deselect_routes
- deselect_secondary_pg_net
- edit_add_polygon_point
- edit_add_route_point
- edit_add_trim_metal
- edit_adjust_routes
- edit_create_via
- edit_cut_route
- edit_duplicate_routes
- edit_end_polygon_point
- edit_end_route_point
- edit_merge_routes
- edit_move_routes
- edit_resize_routes
- edit_rotate
- edit_split_routes
- edit_split_wide_routes
- edit_trim_routes
- edit_update_route_layer
- edit_update_route_mask
- edit_update_route_net
- edit_update_route_rule

- edit_update_route_status
- edit_update_via
- edit_wire_via Category Attributes
- edit_wire Category Attributes
- end_parallel_edit
- read_parallel_edit_files
- report_legal_wire_width_for_bump
- select_routes
- select_secondary_pg_net
- start_parallel_edit

convert_special_routes_for_eco

```
convert_special_routes_for_eco
[-help]
[-exclude_nets nets]
[-nets nets]
```

Converts special nets to geometrically equivalent regular nets. You can convert all special nets, named special nets, or all special nets except those you specify. The `convert_special_routes_for_eco` command supports polygon shapes as well.

Notes

- If you do not specify any parameters, this command converts all special nets to signal nets.
- This command does not work for PG nets.

Parameters

`-help`	Prints a brief description that includes type and default information for each `convert_special_routes_for_eco` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man convert_special_routes_for_eco`
`-exclude_nets` *nets*	Specifies nets you do not want to convert. You can use wildcards. • If a name contains a special character, enclose it in braces (`{}`). • If you specify multiple net names, enclose the names in braces (`{}`). *Data_type*: (net)*, optional
`-nets` *nets*	Specifies the nets to convert. You can use wildcards. • If a name contains a special character, enclose it in braces (`{}`). • If you specify multiple net names, enclose the names in braces (`{}`). *Data_type*: (net)*, optional

Note: For backward compatibility, this command also supports the following syntax for nets to convert:

`convert_special_routes_for_eco` *netName* [*netName2 netName3 ...*]

Examples

- The following command converts special nets named `clk*` to regular nets:
  `convert_special_routes_for_eco -nets clk*`
- The following command converts all special nets, except those named `clk1*`, to regular nets:
  `convert_special_routes_for_eco -exclude_nets clk1*`
- The following command converts special nets named `clk*` to regular nets, except nets named `clk1*`:
  `convert_special_routes_for_eco -nets clk* -exclude_nets clk1*`
- The following command converts all special nets named `clk*`, `a2`, and `s2[1]`, except for nets `clk1` and `clk2`, to regular nets:
  `convert_special_routes_for_eco -nets {clk* a2 s2[1]} -exclude_nets {clk1 clk2}`

Related Information

- `gui_undo`
- `gui_redo`
- Attributes in the `edit_wire` category
- Editing Wires chapter in the *User Guide*
- Edit Menu chapter in the *Menu Reference*
    - *Wire* section

create_fillet

```
create_fillet [-help]
```

Adds a tear-drop fillet between a wire and bump connection. A teardrop fillet is used to reinforce the mechanical strength of the connection to the bump. This applies to all bumps in the design. If the connected wire is too short to create the fillet, a marker is created to indicate the failure.

Parameters

`-help`	Prints a brief description that includes type and default information for each `create_fillet` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man create_fillet`

Related Information

- `delete_fillet`
- `gui_undo`
- `gui_redo`
- Attributes in the `edit_wire` category
- Editing Wires chapter in the *User Guide*
- Edit Menu chapter in the *Menu Reference*
  - *Wire* section

delete_fillet

```
delete_fillet [-help]
```

Deletes the tear-drop fillet that was added between a wire and bump connection with the `create_fillet` command.

Parameters

`-help`	Prints the usage of the command. For a detailed description of the command, use the `man` command: `man delete_fillet`

Related Information

- `create_fillet`
- `gui_redo`
- `gui_undo`
- Attributes in the `edit_wire` category
- Editing Wires chapter in the *User Guide*
- Edit Menu chapter in the *Menu Reference*
  - *Wire* section

delete_routes

```
delete_routes
[-help]
[-area {x1 y1 x2 y2}]
[-direction {h | v | 45 | 135}]
[-floating_via]
[-layer {list_of_layers}]
[-net net*]
[-parallel_routing]
[-physical_pin_only]
[-obj_type {wire via}]
[-regular_wire_with_drc]
[-selected]
[-shapes  {ring stripe followpin iowire corewire blockwire padring blockring fillwire fillwireopc
drcfill none}]
[-shield]
[-shield_only]
[-status {fixed noshield routed shield unknown}]
[-sub_class sublass_name]
[-trim_metal]
[-type {regular | special | patch}]
[-use {clock | power | signal} ]
[-via_cell cell_name_list]
[-with_edit_flag {0 | 1}]
```

Deletes physical wires, any vias connected to the wires, and violation markers associated with the wires.
Use `delete_routes -physical_pin_only` to delete physical pins in the design.

By default, when you delete a wire or via with delete_routes, the tool searches and deletes all related DRC markers. If you do not want the related DRC markers to be cleared when you delete a wire or via, set the `edit_wire_drc_on` root attribute to `false` before using `delete_routes`. This can improve the run time if have to delete a large number of wires and vias.

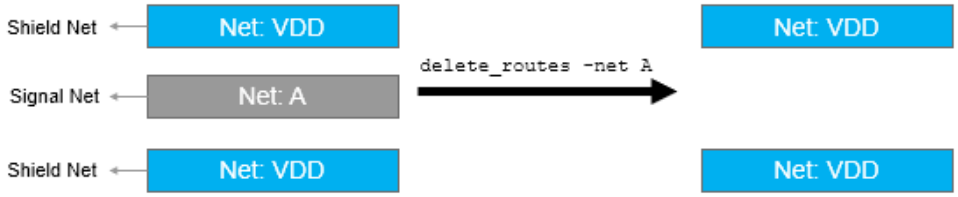
By default, the `delete_routes` command retains TSVs and deletes only the real routing layers and vias. You can change this behavior by setting the `edit_wire_delete_tsv` parameter to `true` before using `delete_routes`.

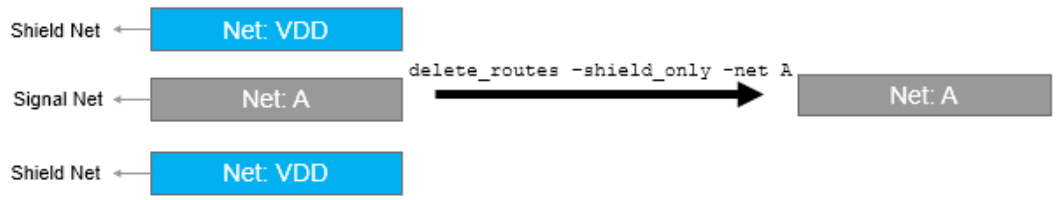
Note: The `delete_routes` command deletes only those vias that are connected to wires. To delete vias that are not connected to wires, use the `delete_metal_fill` command.

The `delete_routes` command deletes physical wires. The `delete_nets` command deletes logical nets. For more information, see `delete_nets`.

Parameters

`-help`	Prints a brief description that includes type and default information for each `delete_routes` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man delete_routes`
`-area {x1 y1 x2 y2}`	Specifies the coordinates for the area from which to delete the wires. *Default:* Deletes all wires. *Data_type*: rect, optional
`-direction {h | v | 45 | 135}`	Specifies the orientation of the wires to delete. *Default:* If you do not specify this parameter, deletes wires with either orientations. `h`: Specifies that only horizontal wires are to be deleted. `v`: Specifies that only vertical wires are to be deleted. `45`: Specifies that only 45-degree wires are to be deleted. `135`: Specifies that only 135-degree wires are to be deleted. *Data_type*: enum, optional
`-floating_via`	Specifies that only floating vias should be deleted. *Data_type*: bool, optional
`-layer {list_of_layers}`	Specifies the layers from which to delete wires. You must specify layers by using their LEF names. You must use LEF layer names or a routing layer index value (1 for the first routing layer, 2 for the second routing layer, and so on). This option supports backside and Through Silicon Via (TSV) layers as well. *Default:* If you do not specify this parameter, deletes wires from all layers. *Data_type*: string, optional

`-net net*`	Specifies the net or list of nets from which to delete wires. You can use wildcard values in this field. For example, if you specify `V*`, deletes wires from nets `VDD` and `VSS`. If you are specifying multiple nets, enclose them in braces. **Note**: By default, the wires and vias shielding the nets specified with `delete_routes -net` are not deleted. If you want to delete the shielding as well, specify `-shield` along with `-net`.  *Data_type*: (net)*, optional
`-obj_type {wire via}`	Specifies whether wires or vias should be deleted. You can specify both as well. If `-obj_type` is not specified, `editDelete` deletes both wires and vias by default. *Data_type*: (enum)+, optional
`-parallel_routing`	Specifies that the wires created by NanoRoute parallel routing are to be deleted. *Data_type*: bool, optional
`-physical_pin_only`	Specifies that the physical pins in the design are to be deleted. You can use filter options, such as `-net`, `-layer`, `-area`, and `-selected` along with `-physical_pin_only` to delete specific physical pins. *Data_type*: bool, optional
`-regular_wire_with_drc`	Deletes the regular wires with violation markers. This parameter works only when `-net` or `-status` are specified as filters. *Data_type*: bool, optional
`-selected`	Deletes selected wires, vias, and physical pins. **Note**: Only physically selected objects are deleted. If you want to delete all wires from a net, use `select_routes -net` to ensure the wires of the net are physically selected. *Data_type*: bool, optional
`-shapes {ring stripe followpin iowire corewire blockwire padring blockring fillwire fillwireopc drcfill none}`	

	Specifies the shapes of the wires to delete. The status must be in all upper-case letters. If you specify multiple shapes, enclose the shapes in braces. For shape definitions, see *Shapes,* in the "DEF Syntax" chapter of the LEF/DEF Language Reference. *Default:* If you do not specify this parameter, deletes wires of all shapes. *Data_type*: (enum)+, optional
`-shield`	Deletes the wires and vias of the nets specified with the `-net` parameter, along with the wires and vias shielding the specified nets. **Note**: This parameter cannot be used with `-shield_only`. *Data_type*: bool, optional
`-shield_only`	Deletes only the wires and vias shielding the nets specified with the `-net` parameter. The signal nets being shielded are retained.  **Note**: This parameter cannot be used with `-shield`. *Data_type*: bool, optional
`-status {fixed noshield routed shield unknown}`	
	Specifies the status of wires to delete. If you specify more than one status, enclose the status list in curly braces. **Note:** If you select `-type special -status noshield` or `-type regular -status shield` the software generates an error message. *Data_type*: (enum)+, optional
`-sub_class` *`sublass_name`*	Specifies the subclass name of the wires to be deleted. Use this parameter if you want to delete rings, stripes, or other wires belonging to a specific subclass. **Note:** You can use wildcards in *`subclass_name`*. Only special wire is supported by this option. *Data_type*: string, optional

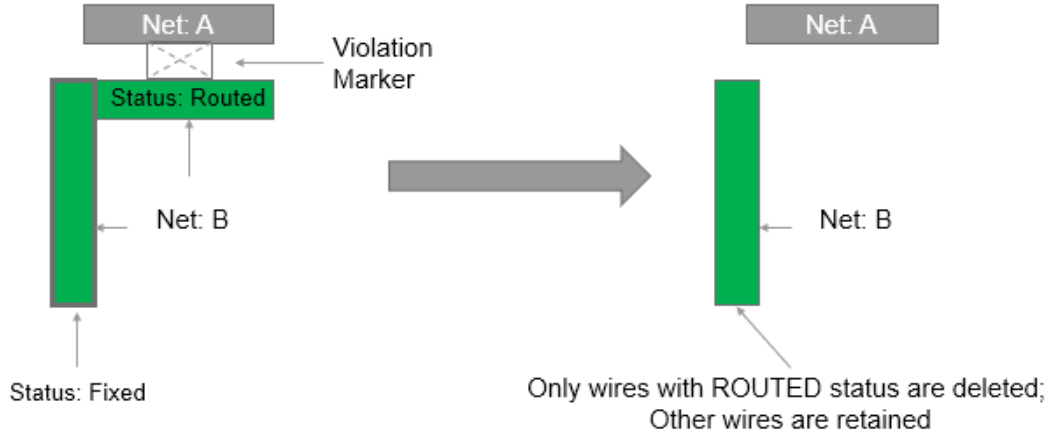
`-trim_metal`	Deletes the trim metal attached to all patch wires, without deleting the patch wires. If `-selected` is also specified, deletes only the trim metal attached to the selected patch wire without deleting the patch wire. *Data_type*: bool, optional
`-type {regular | special | patch}`	Specifies whether to delete regular wires, special wires, or patch wires. **Note:** If you specify `regular`, omit the `-shapes` parameter or specify `-shapes none`. *Data_type*: enum, optional
`-use {clock | power | signal}`	
	Indicates that wires will be deleted on the basis of the specified use: `clock`: Specifies that only wires for clock nets will be deleted. `power`: Specifies that only wires for power nets will be deleted. `signal`: Specifies that only wires for general signal nets will be deleted. *Data_type*: enum, optional
`-via_cell` *cell_name_list*	
	Specifies the via cells of the vias to be deleted. You can use wildcards while specifying the via cell names. If you are specifying multiple cell names, enclose them in braces.
`-with_edit_flag {0 | 1}`	Deletes all modified wires, that is wires with value `1` for the `isEdit` property on the net. **Notes** • As the `isEdit` property is added on the whole net, all the wires on the net with `isEdit` value set to `1` will be deleted. • The `isEdit` property is used to flag nets with modified wires and vias. This is useful when evaluating how many objects need to be modified for fixing DRC violations and then deciding whether to fix the violations manually or through ECO routing. The `isEdit` property is controlled through the `edit_wire_create_is_edit_flag` root attribute in the `edit_wire` category. By default, this attribute is set to `false`. If you set the value of this attribute to `true`, the `isEdit` property is set to `1` for nets with modified wires or vias. *Data_type*: enum, optional

Examples

- The following command removes wires from the *METAL1* and *METAL2* layers:
  ```
  delete_routes -layer {METAL1 METAL2}
  ```
- The following command removes horizontal fillwires from the *METAL1* layer:
  ```
  delete_routes -layer METAL1 -direction h -shapes fillwire
  ```
- The following command removes wires from the specified area:
  ```
  delete_routes -area 100 100 200 200
  ```
- The following command removes the selected wires and vias from the specified area:
  ```
  delete_routes -area 100 100 200 200 -selected
  ```
- The following command only deletes the special wires that belong to the net `carry4` leaving the vias untouched:
  ```
  delete_routes -type special -net carry4 -obj_type {wire}
  ```
- The following command deletes all physical pins:
  ```
  delete_routes -physical_pin_only
  ```
- The following command deletes all physical pins from net VSS:
  ```
  delete_routes -net VSS -physical_pin_only
  ```
- The following command deletes all physical pins from nets VSS and VDD:
  ```
  delete_routes -net {VDD VSS} -physical_pin_only
  ```
- The following command deletes all physical pins from layer METAL2:
  ```
  delete_routes -layer METAL2 -physical_pin_only
  ```
- The following command deletes all physical pins from the specified area:
  ```
  delete_routes -area {26 0 31 4} -physical_pin_only
  ```
- The following set of commands deletes the selected physical pin:
  ```
  select_obj <ptr of layerShape> #Selects the pointer of the layer shape
                                 #belonging to a physical pin
  delete_routes -selected
  ```
- The following command deletes the regular wires with violation markers from only the specified net:
  ```
  delete_routes -net B -regular_wire_with_drc
  ```

- The following command deletes only routed regular wires with violation markers from the specified net:

```
delete_routes -net B -regular_wire_with_drc -status routed
```



Related Information

- edit_add_route_point
- edit_adjust_routes
- edit_cut_route
- edit_duplicate_routes
- edit_end_route_point
- edit_merge_routes
- edit_move_routes
- edit_resize_routes
- edit_rotate
- edit_split_routes
- edit_trim_routes
- gui_redo
- gui_undo
- select_routes
- set_db
- Attributes in the edit_wire category
- Editing Wires chapter in the *User Guide*
- Edit Menu chapter in the *Menu Reference*
    - *Wire* section

delete_secondary_pg_net

```
delete_secondary_pg_net
[-help]
-insts inst+
-net net
```

Deletes Power/Ground (P/G) net wire segments from specified instances. This command provides an easy way delete the routing from specific instance P/G pins to stripes.

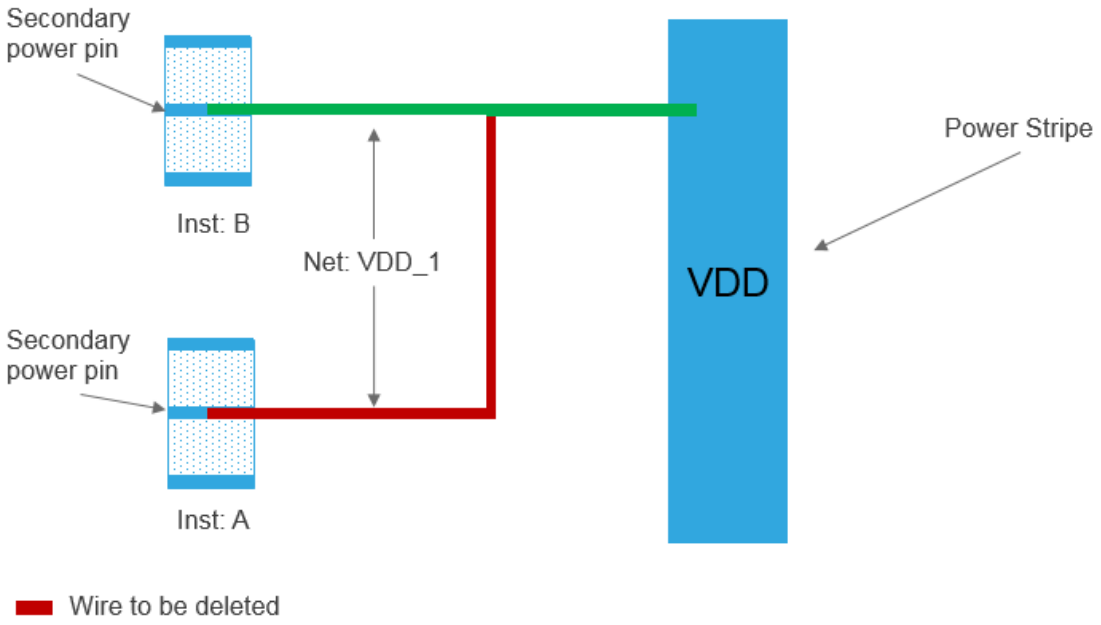
Note: This command does not support dangling secondary P/G nets.

Parameters

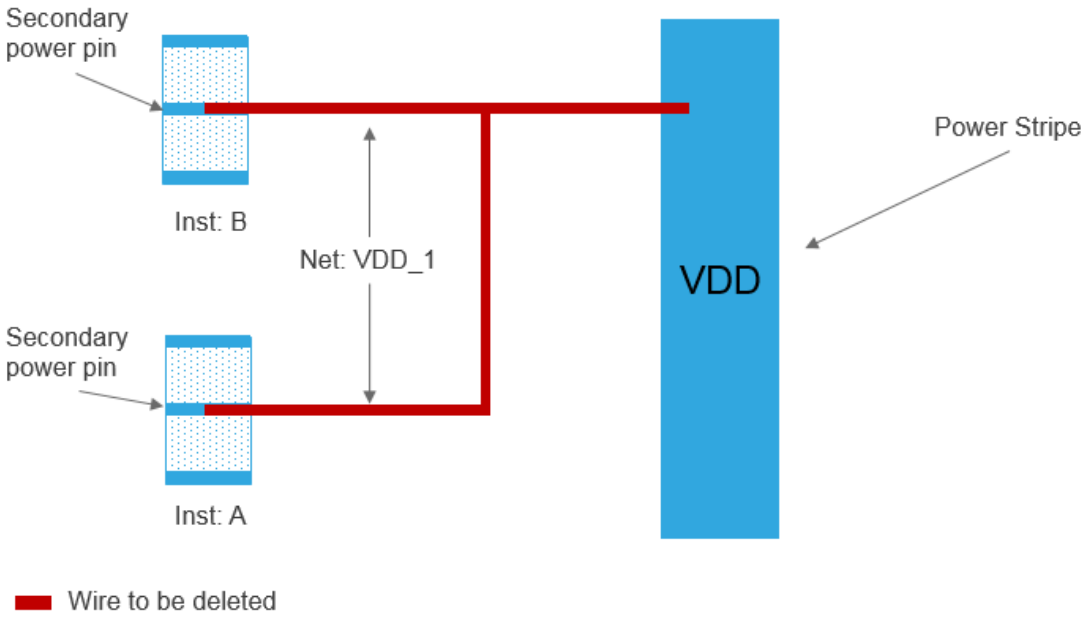
`-help`	Prints the usage of the command. For a detailed description of the command, use the `man` command: `man delete_secondary_pg_net`
`-insts inst+`	Specifies the list of instances that have a secondary PG pin. Wildcards are supported in instance names. If one or more specified instances does not connect with the specified net, the tool displays a warning and ignores those instances. *Data_type*: (inst)+, required
`-net net_name`	Specifies the net name. **Note**: *net_name* does not take any wildcard values. *Data_type*: (net), required

Examples

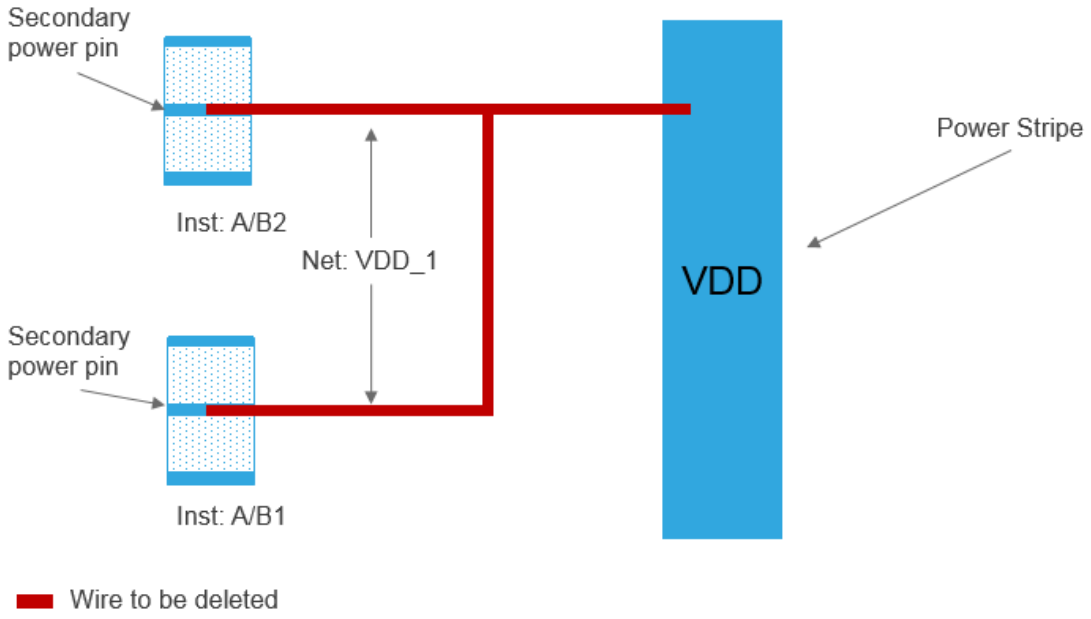
- The following command deletes the routing from the instance `A` P/G pin in net `VDD_1` to the power stripe:
  ```
  delete_secondary_pg_net -net VDD_1 -insts {A}
  ```



- The following command selects the routing from the P/G pins in instances `A` and `B` in net `VDD_1` to the power stripe:
```
delete_secondary_pg_net -net VDD_1 -insts {A B}
```



- The following command selects the routing from the P/G pins in instances `A/B1` and A/B2 in net `VDD_1` to the power stripe:
```
delete_secondary_pg_net -net VDD_1 -insts {A/*}
```



Related Information

- `deselect_secondary_pg_net`
- `select_secondary_pg_net`
- Editing Wires chapter in the *User Guide*
- Edit Menu chapter in the *Menu Reference*
  - *Wire* section

deselect_routes

```
deselect_routes
[-help]
[-area {x1 y1 x2 y2}]
[-direction {h | v | 45 | 135}]
[-floating_via]
[-layer list_of_layers]
[-nets net*]
[-physical_pin_only]
[-regular_wire_with_drc]
[-shapes {ring stripe followpin iowire corewire blockwire padring blockring fillwire fillwireopc
drcfill none}]
[-shield]
[-shield_only]
[-status {cover fixed noshield routed shield}]
[-sub_class sublass_name]
[-type {regular special patch}]
[-use {clock power signal}]
[-via_cell cell_name_list]
[-with_edit_flag {0 | 1}]
[-obj_type {wire via} ]
[-from_pin pin_name -to_pin pin_name]
```

Deselects wires and vias based on specified parameters. Use this command when nets are selected.

Note - You can use this command to deselect nets by property. For example, to delete all signal nets except clock nets, select all signal nets with `select_routes`, deselect the clock nets with this command, then delete the selected nets with `delete_routes`.

Parameters

`-help`	Prints a brief description that includes type and default information for each `deselect_routes` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man deselect_routes`
`-area {x1 y1 x2 y2}`	Specifies the coordinates for the area from which to deselect wires and vias. *Data_type*: rect, optional

`-direction {h | v | 45 | 135}`	Specifies the orientation of the wires to deselect. *Default:* If you do not specify this parameter, deselects wires in any orientation. • `h`: Specifies that horizontal wires are to be deselected. • `v`: Specifies that vertical wires are to be deselected. • 45: Specifies that 45-degree wires are to be deselected. • 135: Specifies that 135-degree wires are to be deselected. *Data_type*: enum, optional
`-floating_via`	Specifies that only floating vias are to be deselected. A via, or a stack of vias, is floating, if it is not connected to another object, wire, pin, or via, on either its top or bottom metal layer or on both these layers. *Data_type*: bool, optional
`-from_pin` *pin_name*	Specifies the name of the starting pin of the wire to be deselected. The starting pin you specify should be connected to the same net as the ending pin specified with `-to_pin`. If they connect to different nets, the tool reports an error. **Note**: Deselection of wires between specified pins is supported only for signal and clock nets. *Data_type*: string, optional
`-layer` *list_of_layers*	Specifies the layers from which to deselect wires and vias. If you specify multiple layers, enclose the layer names in braces. *Default:* If you do not specify this parameter, deselects wires and vias from all layers. *Data_type*: string, optional
`-nets` *net**	Specifies the net names from which to deselect wires and vias. This option supports backside and Through Silicon Via (TSV) layers as well. **Note:** You can use wildcard values in this field. For example, if you specify `V*`, deselects wires and vias from nets `VDD` and `VSS`. *Data_type*: (net)*, optional
`-obj_type {wire via}`	Specifies whether wires or vias or both should be deselected. If the parameter is not specified, only wires are deselected. *Default*: `wire` *Data_type*: (enum)+, optional

`-physical_pin_only`	Specifies that physical pins are to be deselected. By default, `deselect_routes` does not deselect physical pins. You can use the `-physical_pin_only` parameter along with `-area`, `-layer`, or `-nets` to deselect the physical pins in the specified area, layer, or net, respectively. *Data_type*: bool, optional
`-regular_wire_with_drc`	Deselects regular wires with violation markers. *Data_type*: bool, optional
`-shapes {ring stripe followpin iowire corewire blockwire padring blockring fillwire fillwireopc drcfill none}`	
	Specifies the shapes of the wires and vias to be deselected. If you specify multiple shapes, enclose the list of shapes in braces. For shape definitions, see *Shapes,* in the "DEF Syntax" chapter of theLEF/DEF Language Reference. *Default:* If you do not specify this parameter, deselects wires of all shapes. *Data_type*: (enum)+, optional
`-shield`	Deselects the wires and vias of the nets specified with the `-nets` parameter, along with the wires and vias shielding the specified nets. **Note**: This parameter cannot be used with `-shield_only`. *Data_type*: bool, optional
`-shield_only`	Deselects only the shield wires associated with the nets specified using the `-nets` parameter. The signal nets being shielded are remain selected. **Note**: This parameter cannot be used with `-shield`. *Data_type*: bool, optional
`-status {cover fixed noshield routed shield unknown}`	
	Specifies the status of the wires and vias to be deselected. Use all upper-case letters for the status. If you specify more than one status, enclose the status list in curly braces. **Note:** If you select `-type special -status noshield` or `-type signal -status shield` the software generates an error message. *Data_type*: (enum)+, optional

`-sub_class` *`sublass_name`*	Specifies the subclass name of the wires to be deselected. Use this parameter if you want to deselect wires or vias belonging to a specific subclass. Here, *`subclass_name`* is a user-defined attribute that has been previously defined using the `-user_class` parameter of the `add_split_power_vias`, `add_rings`, `add_stripes`, or `route_special` command. For wires, use this parameter if you want to deselect rings, stripes, or other wires belonging to a specific subclass. **Note:** You can use wildcards in *`subclass_name`*. *Data_type*: string, optional
`-to_pin` *`pin_name`*	Specifies the name of the ending pin of the wire to be deselected. The ending pin you specify should be connected to the same net as the starting pin specified with `-from_pin`. If they connect to different nets, the tool reports an error. In addition, the from and to pins you specify should be connected to the same wire. If there there are more than one wire or via between the specified starting and ending pins, the command is not able to deselect the wire path and returns an error. The `-from_pin` and `-to_pin` parameters are compatible with only the `-direction`, `-layer`, `-object_type`, and `-status` options of the `deselect_routes` command. If you use `-from_pin` and `-to_pin` with any other `deselect_routes` parameters, the tool returns an error. **Note**: Deselection of wires between specified pins is supported only for signal and clock nets. *Data_type*: string, optional
`-type {regular special patch}`	Specifies whether to deselect regular wires and vias, special wires and vias, or patch wires. **Note:** If you specify `regular`, you must omit the `-shapes` parameter or specify `-shapes none`. *Data_type*: enum, optional
`-use {clock power signal}`	Specifies whether to select only CLOCK wires/vias, POWER wires, or SIGNAL wires. *Data_type*: enum, optional
`-via_cell` *`cell_name_list`*	Specifies the via cells of the vias to be deselected. You can use wildcards while specifying via cell names. If you are specifying multiple cell names, enclose them in braces. *Data_type*: string, optional

`-with_edit_flag {0 | 1}`	Deselects all modified wires and vias, that is wires and vias with value `1` for the `isEdit` property on the net. **Notes** • As the `isEdit` property is added on the whole net, all the wires and vias on the net with `isEdit` value set to `1` will be deselected. • The `isEdit` property is used to flag nets with modified wires and vias. This is useful when evaluating how many objects need to be modified for fixing DRC violations and then deciding whether to fix the violations manually or through ECO routing. The `isEdit` property is controlled through the `edit_wire_create_is_edit_flag` root attribute in the `edit_wire` category. By default, this attribute is set to `false`. If you set the value of this attribute to `true`, the `isEdit` property is set to `1` for nets with modified wires or vias. *Data_type*: enum, optional

Examples

- The following command deselects all horizontal fillwires on the *METAL1* layer:
  ```
  deselect_routes -layers METAL1 -direction h -shapes fillwire
  ```
- The following command deselects all vias:
  ```
  deselect_routes -obj_type via
  ```
- The following command deselects all wires:
  ```
  deselect_routes
  ```
- The following command deselects all wires and vias:
  ```
  deselect_routes -obj_type {wire via}
  ```

Related Information

- `gui_redo`
- `gui_undo`
- `select_routes`
- `set_db`
- Attributes in the `edit_wire` category
- Editing Wires chapter in the *User Guide*
- Edit Menu chapter in the *Menu Reference*
  - *Wire* section

deselect_secondary_pg_net

```
deselect_secondary_pg_net
[-help]
-insts inst+
-net net
```

Deselects Power/Ground (P/G) net wire segments from specified instances. This command provides an easy way to deselect the routing from specific instance P/G pins to stripes.

Note: This command does not support dangling secondary P/G nets.

Parameters

`-help`	Prints the usage of the command. For a detailed description of the command, use the `man` command: `man deselect_secondary_pg_net`
`-insts inst+`	Specifies the list of instances that have a secondary PG pin. Wildcards are supported in instance names. If one or more specified instances does not connect with the specified net, the tool displays a warning and ignores those instances. *Data_type*: (inst)+, required
`-net net_name`	Specifies the net name. **Note**: *net_name* does not take any wildcard values. *Data_type*: (net), required

Examples

- The following command deselects the routing from the P/G pin in instance `A` of net `VDD_1` to the power stripe:
  ```
  deselect_secondary_pg_net -net VDD_1 -insts {A}
  ```
- The following command deselects the routing from the P/G pins in instances `A` and `B` in net `VDD_1` to the power stripe:
  ```
  deselect_secondary_pg_net -net VDD_1 -insts {A B}
  ```
- The following command deselects the routing from the P/G pins in instances `A/B1` and A/B2 in net `VDD_1` to the power stripe:
  ```
  deselect_secondary_pg_net -net VDD_1 -insts {A/*}
  ```

Related Information

- `delete_secondary_pg_net`
- `select_secondary_pg_net`
- Editing Wires chapter in the *User Guide*
- Edit Menu chapter in the *Menu Reference*
  - *Wire* section

edit_add_polygon_point

```
edit_add_polygon_point
[-help]
point
```

Defines a vertex or corner of a polygon. By issuing the `edit_add_polygon_point` command `n` times, you can create a polygon with `n+1` sides. You can then specify the `edit_end_polygon_point` command to end the polygon at the specified coordinates. `edit_add_polygon_point` honors the `edit_wire_shape` root attribute. This means if you want to create a polygon of a specific shape, you can first use `edit_wire_shape` to specify the required shape and then use `edit_add_polygon_point` and `edit_end_polygon_point` to create the polygon.

The `edit_add_polygon_point` command is DRC aware. When you create a polygon shape using this command, the tool marks any DRC violations in the shape during creation, provided that `edit_wire_drc_on` is set to `true` (default).

Parameters

`-help`	Prints a brief description that includes type and default information for each `edit_add_polygon_point` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man edit_add_polygon_point`
`point`	Specifies the `x` and `y` coordinates of a corner of a wire polygon. *Data_type*: point, required

Example

- The following commands create three-sided wire polygon that ends at the point specified by `edit_end_polygon_point`.

  ```
  edit_add_polygon_point 700 300
  edit_add_polygon_point 700 500
  edit_end_polygon_point 200 400
  ```

  The first side of the wire polygon starts at `700, 300` and ends at `700, 500`. The second side starts at `700, 500` and ends at `200, 400`. The third side is formed by joining the starting point of the polygon (`700,300`) and the ending point specified by `edit_end_polygon_point` (`200,400`).

Related Information

- `edit_end_polygon_point`
- `gui_redo`
- `gui_undo`
- Attributes in the `edit_wire` category
- Editing Wires chapter in the *User Guide*
- Edit Menu chapter in the *Menu Reference*
  - *Wire* section

edit_add_route_point

```
edit_add_route_point
[-help]
point
```

Creates a wire segment that starts or ends at the specified coordinates. Splits diagonal wires whose width is greater than the LEF `MAXWIDTH` parameter for that layer as they are drawn.

If required, the tool allows you to change the net order while adding wire segments. Check the **Examples** sections for details.

If you do not want to view flight lines between Power and Ground (PG) wires and pins when drawing PG nets, use `set_layer_preference flightLine -is_visible 0` to turn off the display of these flightlines. `edit_add_route_point` as well as `edit_end_route_point` honor this setting.

Parameters

`-help`	Prints a brief description that includes type and default information for each `edit_add_route_point` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man edit_add_route_point`
`point`	Specifies the `x` and `y` coordinates for the start point or end point of the wire segment. Note that if the points you specify with each successive `edit_add_route_point` command are not aligned vertically or horizontally, the tool adjusts the coordinates to align the end point of one segment with the start point of the next segment. *Data_type*: point, required

Examples

- The following commands create wire segments that start and stop at the specified points. The complete wire ends at the point specified by `edit_end_route_point`.

```
edit_add_route_point 33.392 36.831    #Start of first wire segment
edit_add_route_point 32.985 32.766    #End of first wire segment, start of second
edit_add_route_point 18.064 32.115    #End of second wire segment
edit_add_route_point 20.951 27.073    #End of third wire segment
edit_end_route_point 20.992 27.114 #End of wire
```

- Use the following set of commands to add wire segments with different orders of nets:

```
gui_set_tool addWire
set_db edit_wire_layer_horizontal M10
set_db edit_wire_nets { a b c d e f}
set_db edit_wire_wire_override_spec {{1 0.062 0.128} {2 0.062 0.1900} {3 0.062 0.128} {4 0.062
0.1900} {5 0.062 0.128} {6 0.062 0.1900} }
edit_add_route_point {1000 2500}
edit_add_route_point {1100 2500}

set_db edit_wire_layer_vertical M9
#Change Net Order
set_db edit_wire_nets { a c b d f e}
set_db edit_wire_wire_override_spec {{1 0.082 0.228} {2 0.082 0.2900} {3 0.082 0.228} {4 0.082
0.6680} {5 0.082 0.228} {6 0.082 0.2900} }
edit_add_route_point {1100 2600}
edit_end_route_point {1100 2600}
```

Related Information

- `edit_end_route_point`
- `gui_redo`
- `gui_set_tool`
- `gui_undo`
- `set_db`
- `set_layer_preference`
- Attributes in the `edit_wire` category
- Editing Wires chapter in the *User Guide*
- Edit Menu chapter in the *Menu Reference*
  - *Wire* section

edit_add_trim_metal

```
edit_add_trim_metal
[-help]
[-layer string]
-location {x y}
[-mask integer]
[-snap_to_end]
```

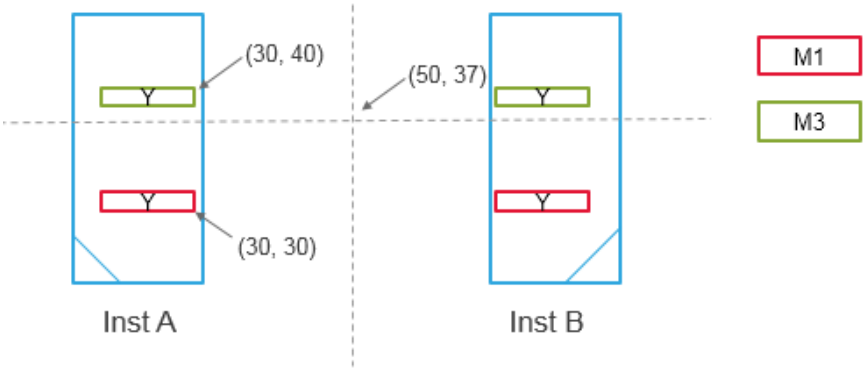
Adds trim metal at the specified location.

Parameters

`-help`	Prints a brief description that includes type and default information for each `edit_add_trim_metal` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man edit_add_trim_metal`
`-layer string`	Specifies the layer on which the trim metal is to be added. The layer name should be a routing layer defined in the LEF file. Note that you can specify only one valid layer name. That is, a layer name list is not supported. If the `-layer` option is not specified, `editAddTrimMetal` adds the trim metal on the nearest object. Check the **Examples** section below to see how to use this option. *Data_type*: string, optional
`-location {x y}`	Specifies the coordinates for the center point of the trim metal. The tool may adjust automatically to the closest location that is best for the trim metal. *Data_type*: point, required
`-mask integer`	Specifies the mask value for the trim metal. However, if `USEMETALMASK` is already defined for the trim layer in the tech LEF, you cannot define the mask value here. *Data_type*: int, optional
`-snap_to_end`	Specifies that the trim metal should be snapped to the closest end of the selected pin or wire. *Data_type*: bool, optional

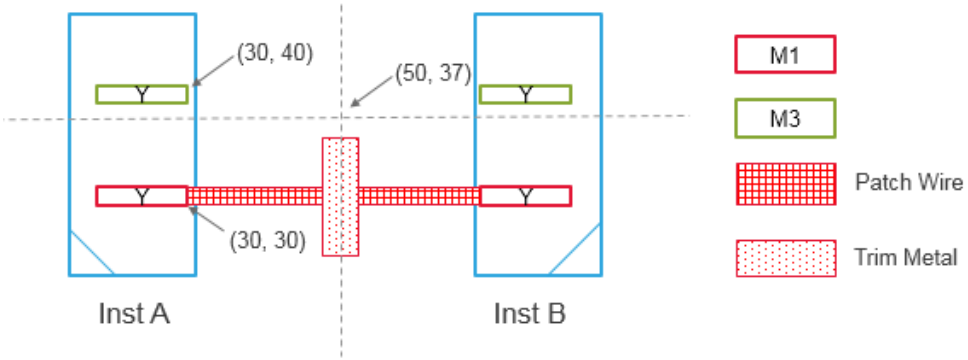
Examples

In the following illustration, the pin Y has two pin shapes, one on `M1` and the other on `M3`. The Pin Y on Inst A is selected.



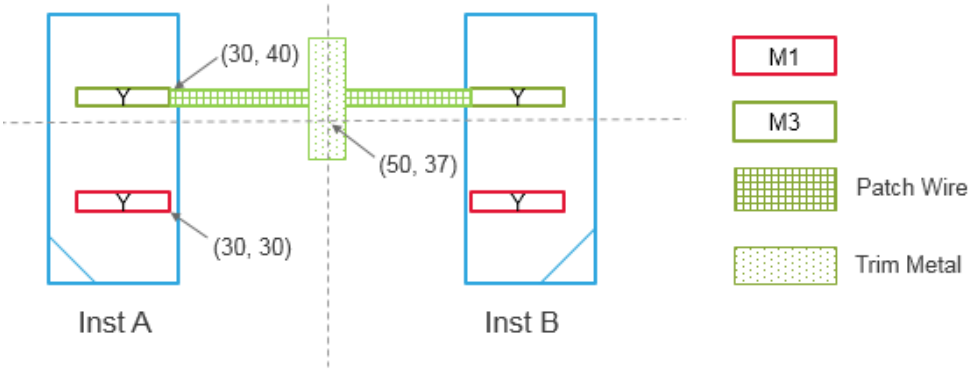
- If you add trim metal with the `-layer` option (as specified below), the tool will add the trim metal on the pin shape that is on layer `M1`.

  ```
  edit_add_trim_metal -layer M1 -location {50 37}
  ```



- If you add trim metal without the `-layer` option (as shown in the command below), the tool will add the trim metal on the nearest pin, which in this case is on `M3`.

  ```
  edit_add_trim_metal -location {50 37}
  ```



Related Information

- `delete_routes`
- `gui_redo`
- `gui_undo`
- `set_db`
- Attributes in the `edit_wire` category
- Editing Wires chapter in the *User Guide*
- Edit Menu chapter in the *Menu Reference*
  - *Wire* section

edit_adjust_routes

```
edit_adjust_routes
[-help]
[-adjust_size]
-refer_obj {top bottom left right}
{[-align_edge {left center right top middle bottom diag45_top diag45_middle diag45_bottom diag135_top
diag135_middle diag135_bottom}]
 [-space_mode {horizontal vertical dist_horizontal dist_vertical diag45 diag135 dist_diag45
dist_diag135}
   [-space float_value]]}
```

Aligns wires and pins on the basis of the specified options. The following objects can be aligned using this command:

- Regular, special, and patch wires
- Physical pins
- Macro pins

The `edit_adjust_routes` command enables you to align objects in one or both of the following modes:

- Align by edge: Use the `-align_edge` parameter to align objects with the reference object by the specified edge.
- Align with spacing: Use the `-space_mode` parameter to adjust the spacing between objects.

Parameters

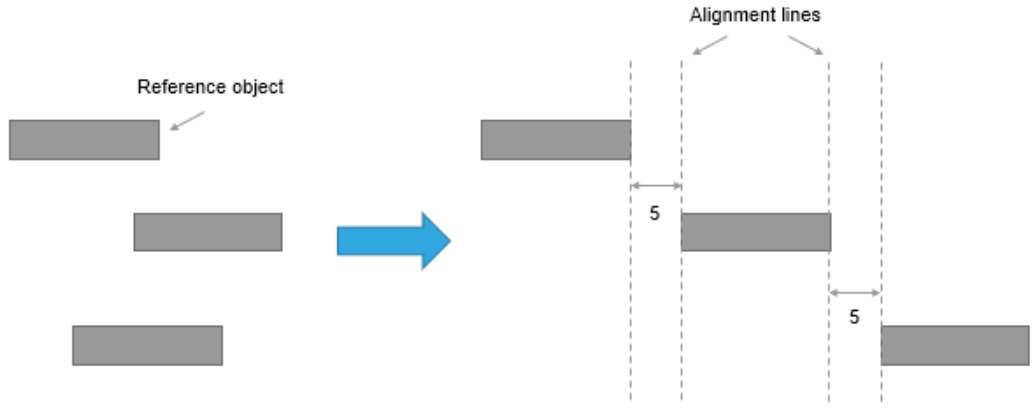
`-help`	Prints a brief description that includes type and default information for each `edit_adjust_routes` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man edit_adjust_routes`
`-adjust_size`	Resizes the objects being aligned to match the reference object. If `-align_edge` is specified, the specified edge of the objects is stretched or shrunk to align with the edge of the reference object. If `-space_mode` is specified, the objects being aligned are resized to be equal to the reference object. *Data_type*: bool, optional
`-align_edge {left center right top middle bottom diag45_top diag45_middle diag45_bottom diag135_top diag135_middle diag135_bottom}`	
	Specifies the edge by which the objects should be aligned with the reference object. *Data_type*: enum, optional
`-refer_obj {top bottom left right}`	

	Specifies the object of reference for alignment. For example, `-refer_obj left` means that objects should be aligned with reference to the leftmost object. This option is mandatory. *Data_type*: enum, required
`-space` *`float_value`*	Specifies the spacing value for aligning objects in the space mode. This option can be specified only if `-space_mode` is specified. *Data_type*: float, optional
`-space_mode {horizontal vertical dist_horizontal dist_vertical diag45 diag135 dist_diag45 dist_diag135}`	
	Specifies that objects should be aligned in the space mode. In this mode, the objects are aligned such that there is equal spacing between them. You can specify one of the following space mode options: • `horizontal` - Aligns the objects such that the horizontal spacing between them is equal to the value specified with `-space`. • `vertical` - Aligns the objects such that the vertical spacing between them is equal to the value specified with `-space`. • `dist_horizontal` - Distributes the objects horizontally. • `dist_vertical` - Distributes the objects vertically. • `diag45` - Aligns the objects in the 45-degree direction such that the spacing between them is equal to the value specified with `-space`. • `diag135` - Aligns the objects in the 135-degree direction such that the spacing between them is equal to the value specified with `-space`. • `dist_diag45` - Distributes the objects in the 45-degree direction. • `dist_diag135` - Distributes the objects in the 135-degree direction. *Data_type*: enum, optional

Example

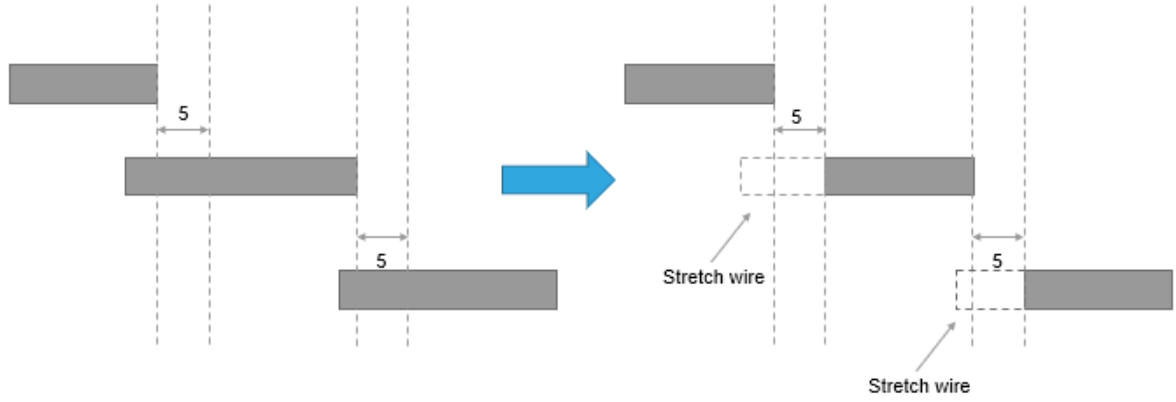
- The following command uses the leftmost object as reference and aligns the wires in the horizontal direction in the space mode with a spacing of 5 units:

```
edit_adjust_routes -space_mode horizontal -spacing 5 -refer_obj left
```



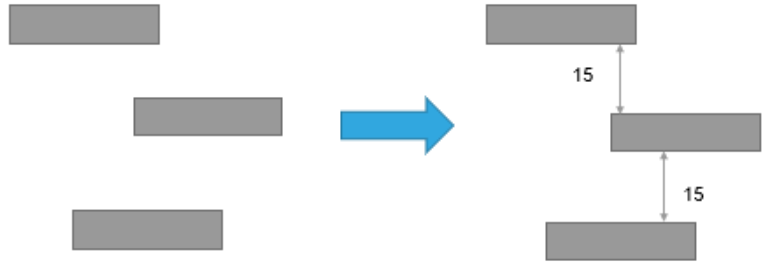
- The following command uses the leftmost object as reference and aligns the wires in the horizontal direction in the space mode with a spacing of 5 units. The command also resizes the objects being aligned to match the reference object:

```
edit_adjust_routes -space_mode horizontal -spacing 5 -refer_obj left -adjust_size
```



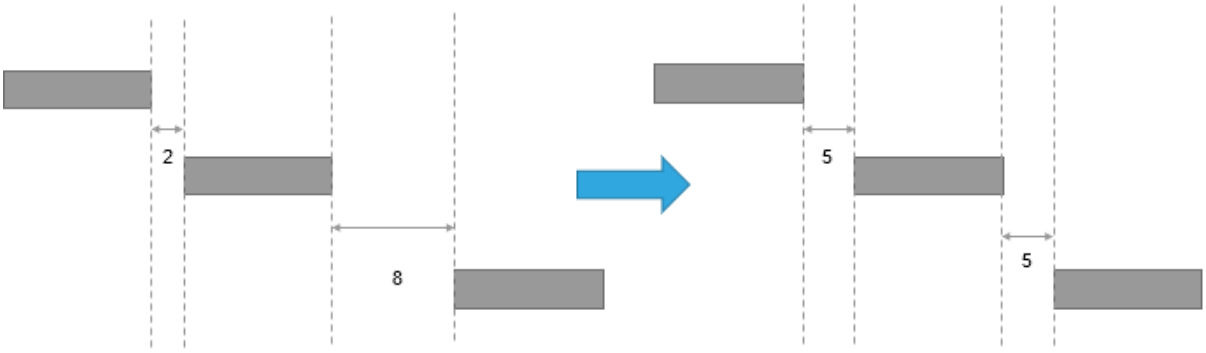
- The following command uses the leftmost object as reference and aligns the wires in the vertical direction in the space mode with a spacing of 15 units:

```
edit_adjust_routes -space_mode vertical -spacing 15 -refer_obj left
```

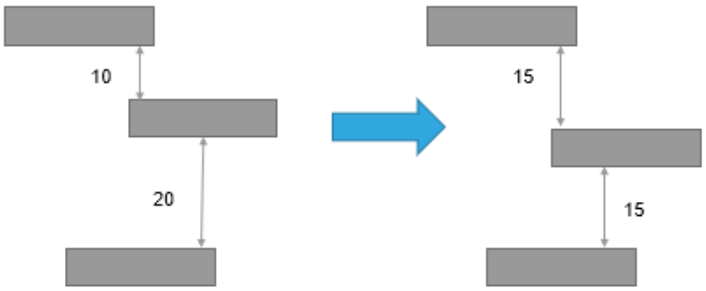


- The following command uses the leftmost object as reference and distributes the wires horizontally:

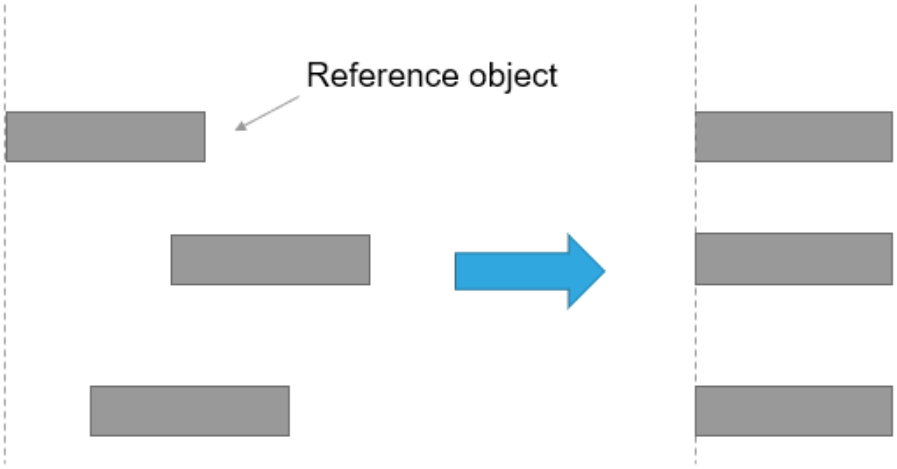
```
edit_adjust_routes -space_mode dist_horizontal -refer_obj left
```



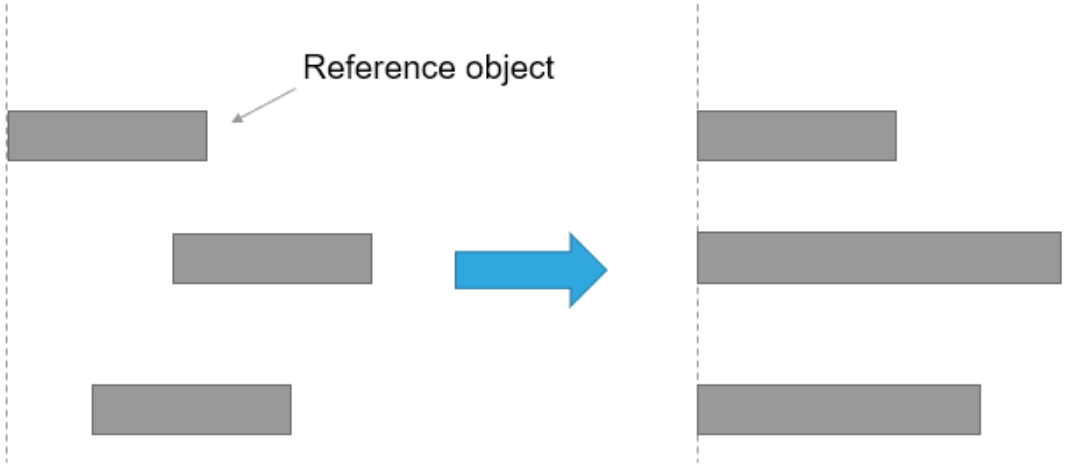
- The following command uses the leftmost object as reference and distributes the wires vertically:
  ```
  edit_adjust_routes -space_mode dist_vertical -refer_obj left
  ```



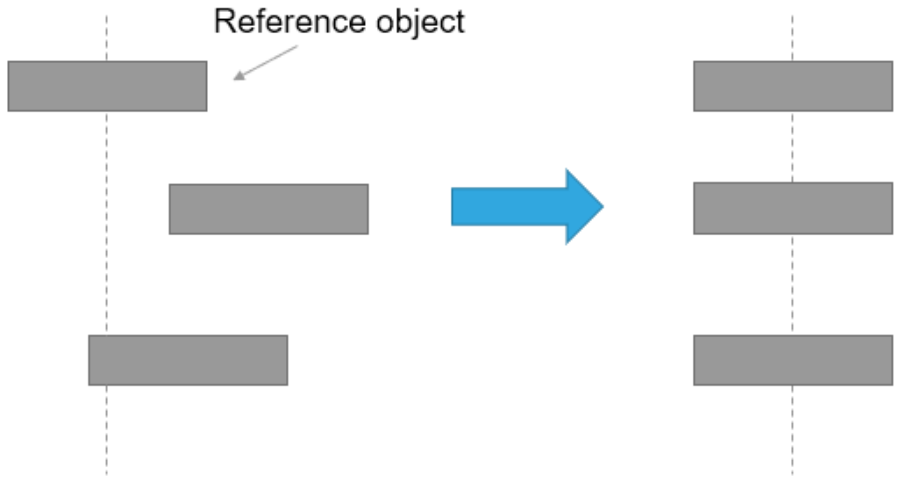
- The following command aligns the left edge of all the wires using the leftmost object as reference:
  ```
  edit_adjust_routes -align_edge left -refer_obj left
  ```



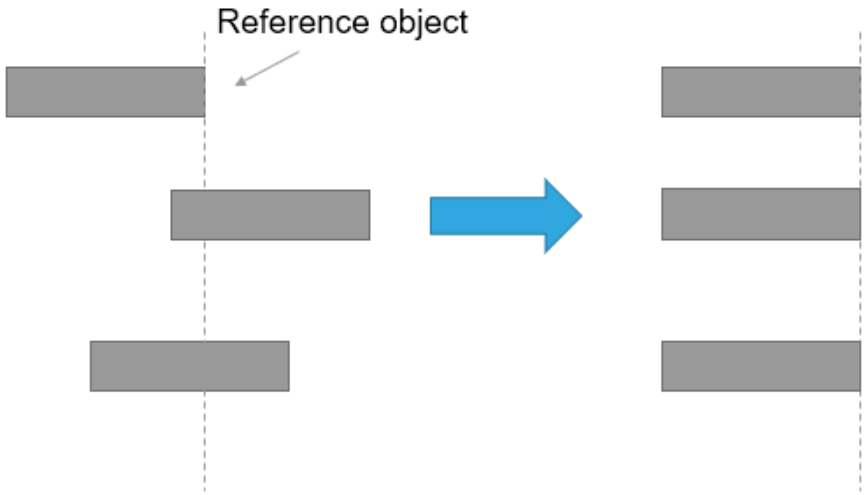
- The following command uses the leftmost wire as reference and resizes the other wires so that their left edges align with the left edge of the reference wire:
  ```
  edit_adjust_routes -align_edge left -refer_obj left -adjust_size
  ```



- The following command aligns the center of all the wires using the leftmost object as reference:
  ```
  edit_adjust_routes -align_edge center -refer_obj left
  ```

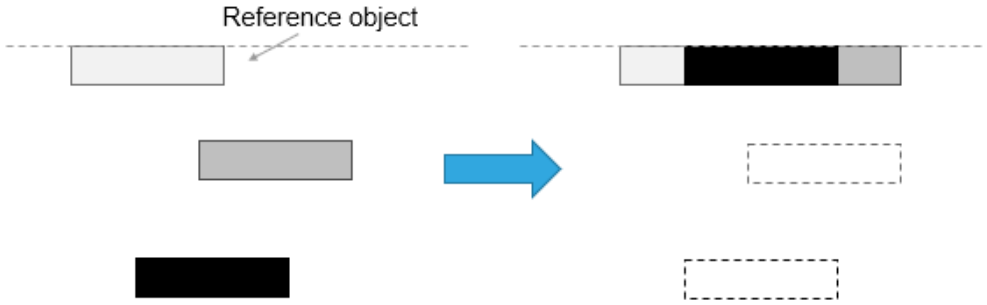


- The following command aligns the right edge of all the wires using the leftmost object as reference:
  ```
  edit_adjust_routes -align_edge right -refer_obj left
  ```



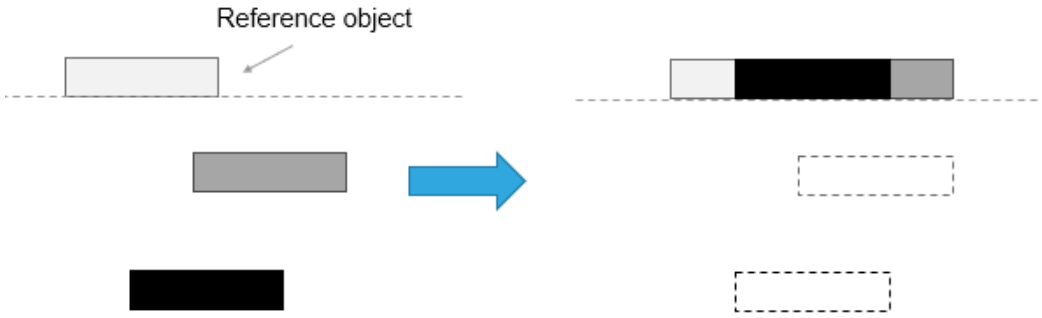
- The following command aligns the top edge of all the wires using the leftmost object as reference:

```
edit_adjust_routes -align_edge top -refer_obj left
```



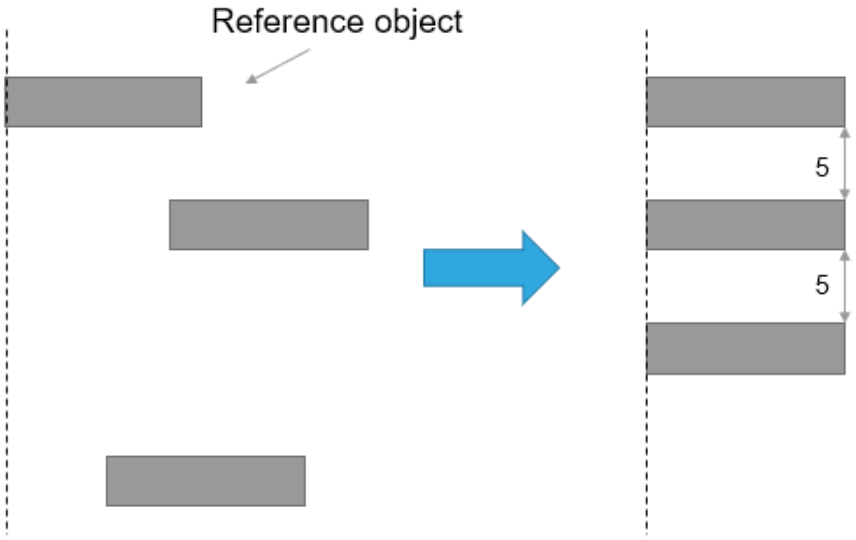
- The following command aligns the bottom edge of all the wires using the leftmost object as reference:

```
edit_adjust_routes -align_edge bottom -refer_obj left
```



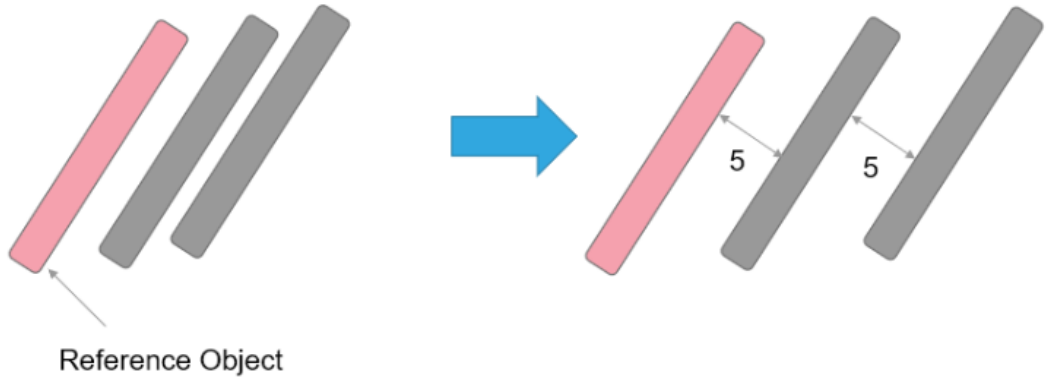
- The following command uses the leftmost object as reference and aligns the left edge of all the wires, keeping a spacing of 5 units between the wires in the vertical direction:

```
edit_adjust_routes -align_edge left -space_mode vertical -spacing 5 -refer_obj left
```



- The following command uses the leftmost object as reference and aligns the wires in the 135-degree direction in the space mode with a spacing of 5 units:

```
edit_adjust_routes -space_mode diag135 -space 5.0 -refer_obj left
```



Related Information

- edit_add_route_point
- edit_cut_route
- edit_duplicate_routes
- edit_end_route_point
- edit_merge_routes
- edit_move_routes
- edit_resize_routes
- edit_rotate
- edit_split_routes
- edit_trim_routes
- delete_routes
- gui_redo
- gui_undo
- set_db
- Attributes in the edit_wire category
- Editing Wires chapter in the *User Guide*
- Edit Menu chapter in the *Menu Reference*
    - *Wire* section

edit_create_via

```
edit_create_via
[-help]
point
```

Creates a via instance and places the origin at the specified coordinates. The origin of the instance's via cell may or may not be at its center. Specify the via parameters with the attributes in the `edit_wire_via` category.

Parameters

`-help`	Prints a brief description that includes type and default information for each `edit_create_via` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man edit_create_via`
point	Specifies the `x` and `y` coordinates of the via origin. *Data_type*: point, required

Example

- The following command adds a via with origin at coordinates `550, 675`:
  ```
  edit_create_via 550 675
  ```

Related Information

- `edit_update_via`
- `gui_redo`
- `gui_undo`
- `set_db`
- Attributes in the `edit_wire` category
- Editing Wires chapter in the *User Guide*
- Edit Menu chapter in the *Menu Reference*
  - *Wire* section

edit_cut_route

```
edit_cut_route
[-help]
[-only_visible_wires]
[-selected]
{-lines {x1 y1 x2 y2 ...} | -box {x1 y1 x2 y2} }
```

Cuts wires, bus guides, or rectangles at the specified location. You can choose to cut by line (using the `-lines` parameter) or by box (using the `-box` parameter. `edit_cut_route` honors `edit_wire_cut_wire_overlap` settings.

Patch wires are also supported.

Parameters

`-help`	Prints a brief description that includes type and default information for each `edit_cut_route` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man edit_cut_route`
`-box {x1 y1 x2 y2}`	Specifies the lower-left and upper-right coordinates of the box for cutting wires, bus guides, or rectangles. The `-box` option currently has the following limitations: • The cut-box must be a rectangle (or square). A polygon shape will be ignored. • Only the wires that are perpendicular to the cut-box edge are cut. In addition, the cut-box edge must pass across the wire, bus guide, or rectangle completely. • This option can be used to cut only horizontal and vertical wires. 45-degree wires are not yet supported. *Data_type*: rect, optional
`-lines {x1 y1 x2 y2 ...}`	Specifies the start and end coordinates of one or more line segments for cutting wires, bus guides, or rectangles. If you specify multiple cut-line segments, they should be parallel to each other. For instance, you cannot specify one horizontal cut-line and one vertical cut-line with the `-lines` parameter. If you do, the tool reports the following error: `Cut lines are not parallel.` In addition, a cut-line must cut across the wire, bus guide, or rectangle

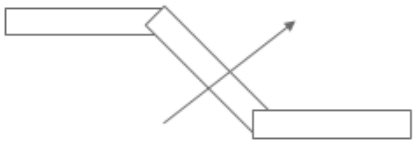
from one edge to the other. Once cut, the signal wire pieces maintain a one-half wire-width extension. Special wires do not maintain an extension.

When you run `edit_cut_route`, the tool snaps the cut-line to the manufacturing grid automatically. The change in coordinates is displayed at the prompt as a warning such as follows:

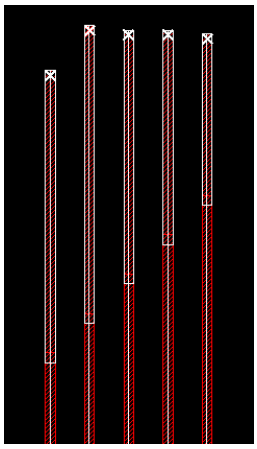
```
WARNING: Cut-line is auto-snapped from (966.343500 1227.245500
966.811500 1227.245500) to (966.345000 1227.245000 966.810000
1227.245000).
```

Note that the tool records only the cut-line coordinate before snapping in the log file.

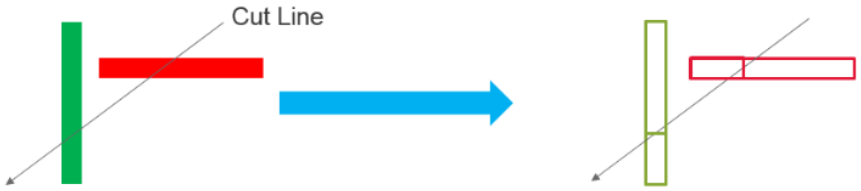
Note: The `-lines` option supports 45-degree wire edits.



In addition, you can also draw a 45-degree cut-line to cut a group of wires in a stair pattern as shown below:

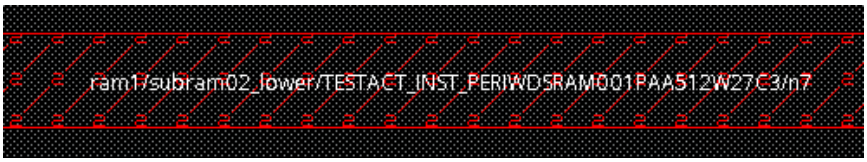


Both regular and special wires can be cut with a 45-degree cut-line. However, patch wires cannot be cut in this way. When you use a 45-degree cut-line, the wires are cut from the touch point of the center line. You can use a single 45-degree cut-line to cut wires in different directions as shown below:

	 *Data_type*: (rect)+, optional
`-only_visible_wires`	Specifies that only the wires on the visible layers are cut. If this parameter is not specified, the tool cuts all the wires overlapping with the cut-line or box. *Data_type*: bool, optional
`-selected`	Specifies that only the selected wires, bus guides, or rectangles should be cut. *Data_type*: bool, optional

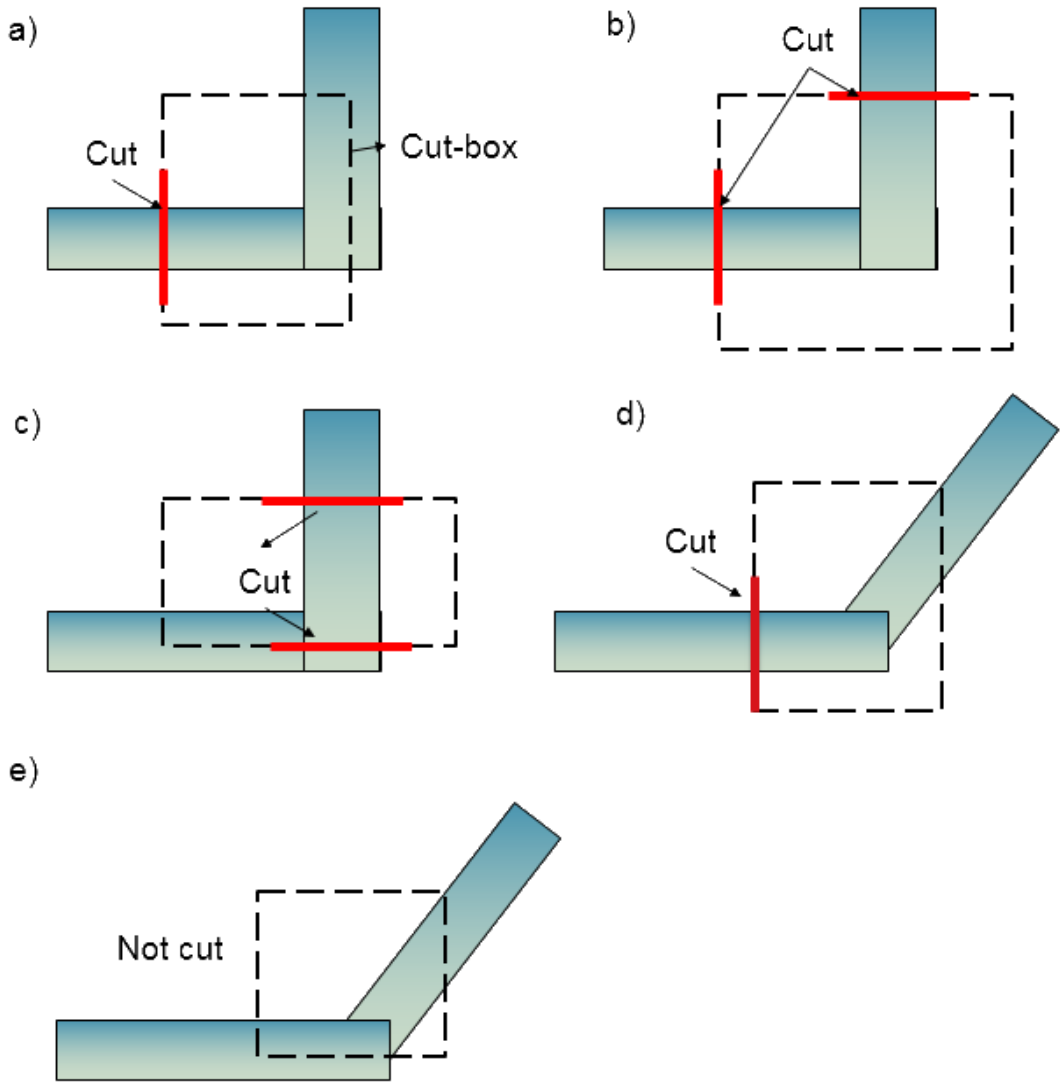
Examples

- The following command cuts the wires through which the specified cut-line passes:
  ```
  edit_cut_route -lines {200 300 200 310}
  ```
- Consider the wire depicted below:

  The following command cuts the wire, along the specified 45-degree cut-line segments, `{95.30 -641.16 95.39 -641.08}` and `{95.69 -641.17 95.78 -641.08}`:
  ```
  edit_cut_route -lines {95.30 -641.16 95.39 -641.08 95.69 -641.17 95.78 -641.08}
  ```


- The following diagrams show examples of using a cut-box to cut wires:
  - In a), the horizontal wire is cut. The vertical wire is not cut because the cut-box edges that are perpendicular to the wire do not pass through it completely.
  - In b), both the horizontal and vertical wires are cut.
  - In c), the vertical wire is cut at the two places where the perpendicular edges of the cut-box pass through it completely. The horizontal wire is not cut.
  - In d), the horizontal wire is cut. The 45-degree wire is not cut because 45-degree wire edits are not yet supported while cutting by box.
  - In e), no cuts are made.


a)
Cut
Cut-box
b)
Cut
c)
Cut
d)
Cut
e)
Not cut

Related Information

- edit_add_route_point
- edit_adjust_routes
- edit_duplicate_routes
- edit_end_route_point
- edit_merge_routes
- edit_move_routes
- edit_resize_routes
- edit_rotate
- edit_split_routes
- edit_trim_routes
- delete_routes
- gui_redo
- gui_undo
- set_db
- Attributes in the edit_wire category
- Editing Wires chapter in the *User Guide*
- Edit Menu chapter in the *Menu Reference*
  - *Wire* section

edit_duplicate_routes

```
edit_duplicate_routes
[-help]
[-layer_horizontal layer]
[-layer_vertical layer]
```

Copies the geometry of selected special wires and pastes them at the same coordinates. If you do not specify a layer, the software pastes them onto the same layer. If you specify a layer, the software copies them to the specified layer and generates vias to maintain connectivity for the duplicated wire.

After duplicating the wires, the software selects the new wires and deselects the original wires. Use the `edit_move_routes` command to move duplicated wires to new locations.

Note: Use the `select_routes` command to select wires.

Parameters

`-help`	Prints a brief description that includes type and default information for each `edit_duplicate_routes` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man edit_duplicate_routes`
`-layer_horizontal` *layer*	Specifies the new layer for duplicated horizontal wires. *Data_type*: string, optional
`-layer_vertical` *layer*	Specifies the new layer for duplicated vertical wires. *Data_type*: string, optional

Example

- The following command duplicates selected horizontal wires and places them on the *METAL3* layer and duplicates selected vertical wires and places them on the *METAL4* layer. It generates vias for the new wires.
  ```
  edit_duplicate_routes -layer_horizontal METAL3 -layer_vertical METAL4
  ```

Related Information

- `edit_add_route_point`
- `edit_adjust_routes`
- `edit_cut_route`
- `edit_end_route_point`
- `edit_merge_routes`
- `edit_move_routes`
- `edit_resize_routes`
- `edit_rotate`
- `edit_split_routes`
- `edit_trim_routes`
- `delete_routes`
- `gui_redo`
- `gui_undo`
- `set_db`
- Attributes in the `edit_wire` category
- Editing Wires chapter in the *User Guide*
- Edit Menu chapter in the *Menu Reference*
    - *Wire* section

edit_end_polygon_point

```
edit_end_polygon_point
[-help]
point
```

Ends a newly created polygon at the specified coordinates. Use this command after at least one `edit_add_polygon_point` command has been issued.

Parameters

`-help`	Prints a brief description that includes type and default information for each `edit_end_polygon_point` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man edit_end_polygon_point`
point	Specifies the `x` and `y` coordinates for the end point of a newly created wire polygon. *Data_type*: point, required

Example

- The following commands create three-sided wire polygon that ends at the point specified by `edit_end_polygon_point`.

  ```
  edit_add_polygon_point 700 300
  edit_add_polygon_point 700 500
  edit_end_polygon_point 200 400
  ```

  The first side of the wire polygon starts at `700, 300` and ends at `700, 500`. The second side starts at `700, 500` and ends at `200, 400`. The third side is formed by joining the starting point of the polygon (`700,300`) and the ending point specified by `edit_end_polygon_point` (`200,400`).

Related Information

- `edit_add_polygon_point`
- `gui_redo`
- `gui_undo`
- Attributes in the `edit_wire` category
- Editing Wires chapter in the *User Guide*
- Edit Menu chapter in the *Menu Reference*
  - *Wire* section

edit_end_route_point

```
edit_end_route_point
[-help]
point
```

Ends a newly created wire segment at the specified coordinates. Splits diagonal wires whose width is greater than the LEF `MAXWIDTH` parameter for that layer as they are drawn. Use this command after at least one `edit_add_route_point` command has been issued.

Parameters

`-help`	Prints a brief description that includes type and default information for each `edit_end_route_point` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man edit_end_route_point`
`point`	Specifies the `x` and `y` coordinates for the end point of a newly created wire segment. *Data_type*: point, required

Example

- The following commands create a wire that ends at the point specified by `edit_end_route_point`:

```
edit_add_route_point 33.392 36.831
edit_add_route_point 32.985 32.766
edit_add_route_point 18.064 32.115
edit_add_route_point 20.951 27.073
edit_end_route_point 20.992 27.114
```

Related Information

- `edit_add_route_point`
- `gui_redo`
- `gui_set_tool`
- `gui_undo`
- `set_db`
- `set_layer_preference`
- Attributes in the `edit_wire` category
- Editing Wires chapter in the *User Guide*
- Edit Menu chapter in the *Menu Reference*
    - *Wire* section

edit_merge_routes

```
edit_merge_routes [-help]
```

Merges all wires of the same width, layer, and net that are colinear with and electrically connected to a selected wire. Does not merge signal wires that contain branches. This command also supports 45-degree wire edits.

This command can also be used to merge selected bus guides.

Note: Use the `select_routes` command to select the wire. Use the `select_obj` command to select bus guides or segments.

Parameters

`-help`	Prints the usage of the command. For a detailed description of the command, use the `man` command: `man edit_merge_routes`

Related Information

- edit_add_route_point
- edit_adjust_routes
- edit_cut_route
- edit_duplicate_routes
- edit_end_route_point
- edit_move_routes
- edit_resize_routes
- edit_rotate
- edit_split_routes
- edit_trim_routes
- delete_routes
- gui_redo
- gui_undo
- select_routes
- set_db
- Attributes in the edit_wire category
- Editing Wires chapter in the *User Guide*
- Edit Menu chapter in the *Menu Reference*
  - *Wire* section

edit_move_routes

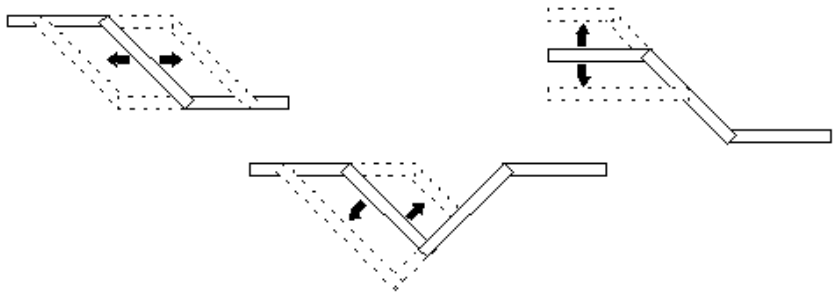
```
edit_move_routes
[-help]
{[[-direction {x y diag45 diag135} -distance double ] | [-dx double -dy double] | -to {x y}] }
```

Moves selected wires or boundary pins.

Moving Wires

`edit_move_routes` moves selected wires in the orthogonal direction. When used with the `-direction` and `-distance` or `-dx` and `-dy` combination, `edit_move_routes` maintains connectivity by stretching connected wires in the orthogonal direction. This command also supports 45 and 135-degree wire edits.

Move 45-degree wire segments while maintaining connectivity



Use the `edit_wire_connect_with_specified_layer` root attribute to control whether layer preference settings are honored when wires are moved. The default value of this parameter is `false`. If you set this parameter to `true`, `edit_move_route` honors the `edit_wire_layer_horizontal` and `edit_wire_layer_vertical` settings while jogging metal layers. For instance, if you cut a vertical wire on `M4` by using `edit_cut_route` such that the wire is divided into three wire segments and then move the wire segment in the middle in the horizontal direction, the tool automatically creates new connecting wires on the layer defined by `edit_wire_layer_horizontal` and drops vias correctly. This means you no longer need to perform the additional step for layer change (`edit_update_route_layer` or `Shift+L`) while moving wires to create a jog. This feature is applicable for both regular and special wires.

To get the current value of the `edit_wire_connect_with_specified_layer` parameter, you can use the `get_db` command.

Note: You can use the `set_db edit_wire_snap_end_to track_regular` and `set_db edit_wire_snap_end_to track_special` commands to specify whether or not to snap wires to the track. To get the current values of these attributes, use `get_db`.

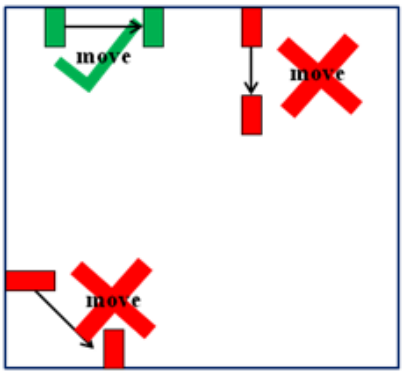
Moving Physical Pins

`edit_move_routes` moves selected physical pins along with connected wires.

Keep the following considerations in mind while trying to move a physical pin with `edit_move_routes`:

- `edit_move_routes` can be used to move only the physical pins on the design boundary. Pins inside the design cannot be moved with this command.

- A pin can be moved only along a boundary.
- It cannot be moved between design boundaries.



- Pins on the top or bottom boundary can be moved only horizontally.
- Pins on the left or right boundary can be moved only vertically.

Note: You can use the `set_db edit_wire_snap_objects_to_track pin` command to snap moved pins to the track. This attribute is set to `true` by default. To get the current value of this attribute, use get_db edit_wire_`snap_objects_to_track pin`.

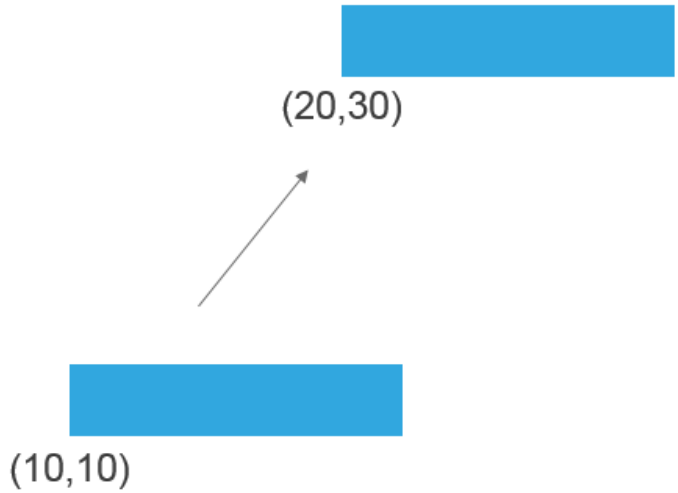
Parameters

`-help`	Prints the usage of the command. For a detailed description of the command, use the `man` command: `man edit_move_routes`
`-direction {x y diag45 diag135}`	
	Specifies the direction of movement. Use one of the following options: • `diag45` - Moves selected wires the specified *`-distance`* in a 45-degree direction. This option cannot be used for moving physical pins. • `diag135` - Moves selected wires the specified *`-distance`* in a 135-degree direction. This option cannot be used for moving physical pins. • `x` - Moves selected vertical wires or pins to the right (if you specify a positive value for *`-distance`*) or left (if you specify a negative value for *`-distance`*). • `y` - Moves selected horizontal wires or pins up (if you specify a positive value for *`-distance`*) or down (if you specify a negative value for *`-distance`*). *Data_type*: enum, optional
`-distance` *`double`*	Specifies the distance to move the wire or pin in microns. *Data_type*: double, optional
`-dx` *`double`*	Specifies the distance in microns by which the selected wire should be moved in the x direction. You must specify the `-dy` value along with it. For example, if you want to move the wire in the x direction by 0.2 microns, specify the following command: `edit_move_routes -dx 0.2 -dy 0` *Data_type*: double, optional
`-dy` *`double`*	Specifies the distance in microns by which the selected wire should be moved in the y direction. You must specify the `-dx` value along with it. For example, if you want to move the wire in the y direction by 0.2 microns, specify the following command: `edit_move_routes -dx 0 -dy 0.2` *Data_type*: double, optional
`-to {x y}`	Specifies the coordinates of the new location to which the selected wire or via is to be moved. If you are moving a via, the `x` and `y`values specify the new coordinates of the via origin. In the case of a wire, the `x` and `y` values specify the new coordinates of the lower left corner of the wire. See Examples. When you move the selected wire or via with `editMove -to`, connected wires are not moved. *Data_type*: point, optional

Examples

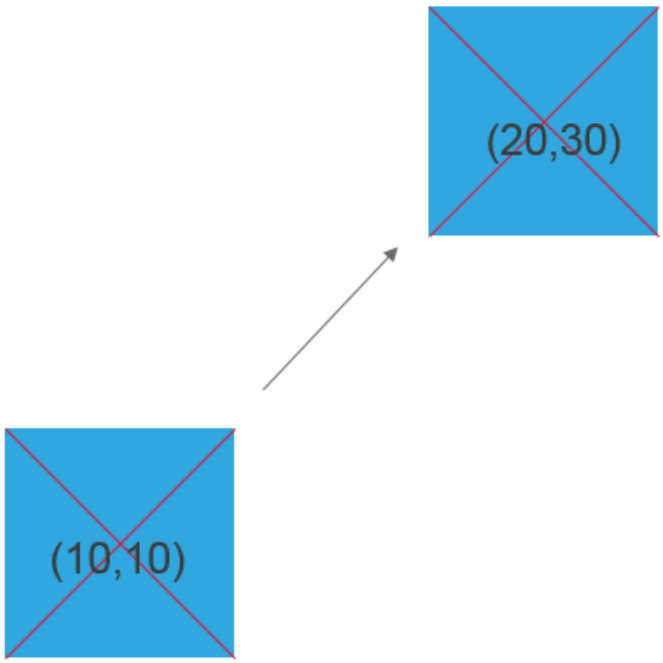
- The following command moves the selected wire such that its lower-left corner lies on the coordinates {`20, 30`} after the move:

```
edit_move_routes -to {20 30}
```



- The following command moves the selected via such that its original lies on the coordinates {`20, 30`} after the move:

```
edit_move_routes -to {20 30}
```



Related Information

- edit_add_route_point
- edit_adjust_routes
- edit_cut_route
- edit_duplicate_routes
- edit_end_route_point
- edit_merge_routes
- edit_resize_routes
- edit_rotate
- edit_split_routes
- edit_trim_routes
- delete_routes
- gui_redo
- gui_undo
- select_routes
- set_db
- Attributes in the edit_wire category
- Editing Wires chapter in the *User Guide*
- Edit Menu chapter in the *Menu Reference*
  - *Wire* section

edit_resize_routes

```
edit_resize_routes
[-help]
[-keep_center_line {0 1 auto}]
[-no_connection]
[-polygon_edge_start {x y} | -to float_value]
{[[-direction {x y diag45 diag135} -side {high low}] | [-polygon_edge_start {x y} -polygon_edge_end
{x y}]] [-offset float_value | -to float_value]] | [-corner {low_left low_right up_left up_right} -
dx float_value -dy float_value]}
```

Resizes wires and pins. The `edit_resize_routes` command can be used to:

- Change the length of wires.
- Change the width of wires.
- Enlarge or shrink wires by changing both the length and width of a wire simultaneously.
- Change the width of physical pins.

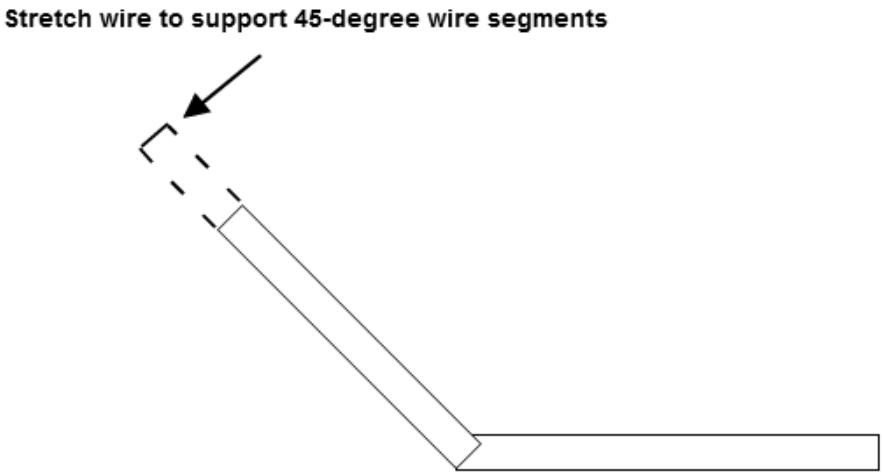
> While changing the wire width to fix EM violations, apply the following setting before invoking the `edit_resize_routes` command:
>
> ```
> set_db edit_wire_unrestricted_regular_wire_width true
> ```
>
> By default, when you are changing the width of a wire, it snaps to the closest non-default route (NDR) available. If you set the `edit_wire_unrestricted_regular_wire_width` attribute to `true`, a wire can be stretched to any width without any restrictions.

Typically, the `edit_resize_routes` command can be used in two modes:

Mode 1 - Change the size of a wire or pin in only one direction at a time

If `-direction` is specified, the tool changes the size in the specified direction only. In this mode, the `edit_resize_routes` command can be used to:

- Change the length of selected regular and special wires by stretching or shrinking them and then snapping them to the closest manufacturing grid point. The `edit_resize_routes` command also supports the stretching/shrinking of 45 and 135-degree wires.



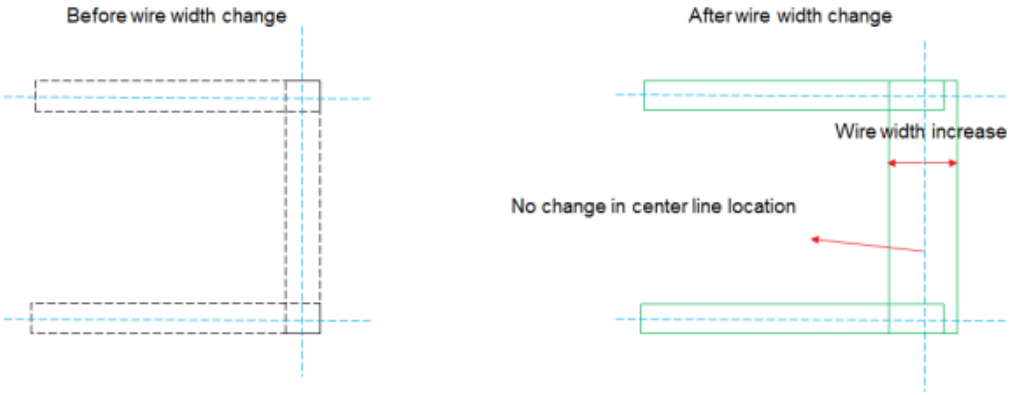
Note: Use the `select_routes` command to select the wires.

- Change the width of selected wires and update vias connected to the changed wires to maintain the via-to-wire dimension ratio. `edit_resize_routes` can be used to change the width of regular as well as special wires:
  - For regular wires, if the width value you specified is not defined in the tech LEF, the tool snaps the width of the regular wire to the closest value defined in the default or non-default rule and prints the following message in the log file:

    ```
    WARNING: The wire width is not defined in default or non default rule, snap wire width to the closest value defined in LEF file
    ```

    The tool also selects a suitable via cell defined in the tech LEF file. If a suitable via cell is not available, the tool does not drop any via.

    When you change the width of a regular wire, the center line is still connected with the neighboring wires after the change:



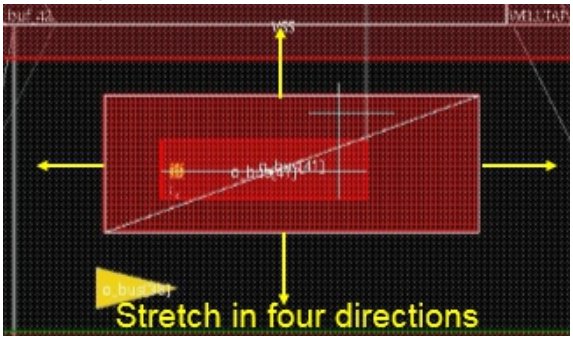
  - For a special wire, you can change the wire width freely by specifying an incremental value. The tool uses viagen to generate a via after the width change of a special wire.

Note: Use the `select_routes` command to select the wires.

If you change the width of a wire, and then change it back to the original width, you might see a discrepancy

due to rounding.
Note: If you specify a width value that is not an even multiple of the manufacturing grid value, the software displays a warning.

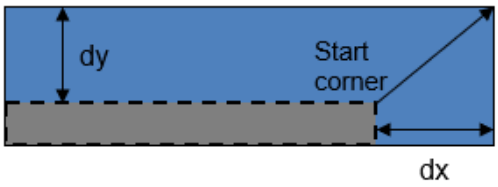
- Resize patch wires. Patch wires can be stretched in four directions:



- Change the width of physical pins. While using the `edit_resize_routes` command to change the pin width, keep the following points in mind:
  - If `set_db edit_wire_drc_on true` has been specified before modifying the pin width, the tool follows the rules from the LEF file while changing pin width.
  - The pin width, whether vertical or horizontal, should not be less than the minimum width defined in the LEF file.
  - If you specify a value less than minimum width for a pin, the tool uses the minimum width for that pin.

Mode 2 - Change the size of a wire in both the x and y directions simultaneously

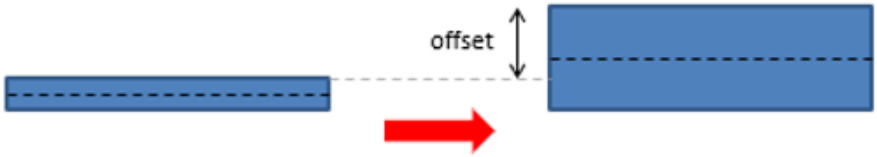
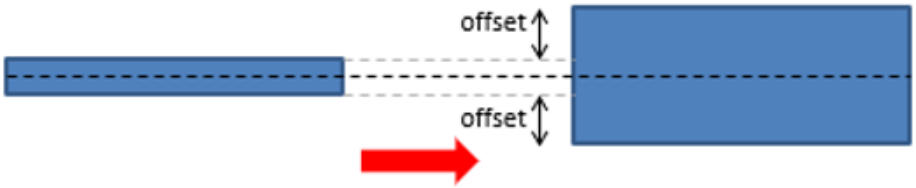
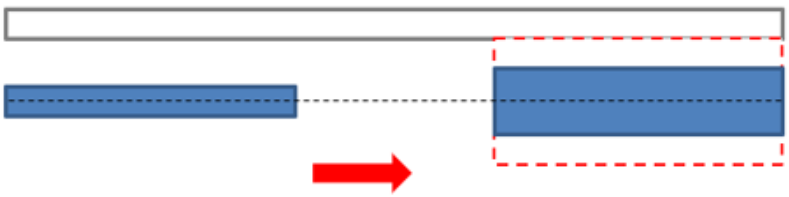
If `-direction` is not specified, the tool changes both the width and height of the selected wire at the same time. You can use `-corner` to specify the start corner from which to stretch or shrink the wire. This mode works for only horizontal or vertical special wires in the no_connection mode. Use `-dx` and `-dy` to specify the distance to stretch the wire in x and y directions, respectively.



Parameters

`-help`	Prints a brief description that includes type and default information for each `edit_resize_routes` parameter. For a detailed description of the command, use the `man` command: `man edit_resize_routes`
`-corner {low_left low_right up_left up_right}`	

	Specifies the starting corner from which the wire should be stretched. *Data_type*: enum, optional
`-direction {x y diag45 diag135}`	
	Specifies the direction in which the wire or pin is to be stretched or shrunk: ◦ `x` - Resizes wires or pins horizontally. For a horizontal wire, specify the direction as `x` if you want to modify the length of the wire (i.e stretch or shrink the horizontal wire). For a vertical wire, specify the direction as `x` if you want to modify the width of the wire. ◦ `y` - Resizes wires or pins vertically. For a horizontal wire, specify the direction as `y` if you want to modify the width of the wire. For a vertical wire, specify the direction as `y` if you want to modify the length of the wire (i.e stretch or shrink the vertical wire). ◦ `diag45` - Resizes the selected wires in the 45-degree direction by the specified offset value. **Note**: This parameter can be used only for wires. ◦ `diag135` - Resizes the selected wires in the 135-degree direction by the specified offset value. **Note**: This parameter can be used only for wires. *Data_type*: enum, optional
`-dx` *`float_value`*	Specifies the distance to which the wire should be stretched in the x direction. *Data_type*: float, optional
`-dy` *`float_value`*	Specifies the distance to which the wire should be stretched in the y direction. *Data_type*: float, optional
`-keep_center_line {0 1 auto}`	

	Controls whether the wire or physical pin is resized with the center line unchanged: • `0` - Resizes the wire or pin from the specified side only. This option works in both the connection and no_connection modes.  This setting cannot be used for changing the width of signal wires. • `1` - Resizes the wire or pin from both the low and high sides, even if you are stretching it from only one side. This is done to keep the center line unchanged.  This option works in both the connection and no_connection modes for wires. However, a physical pin can keep the center line unchanged during a resize in the no-connection mode only. • `auto` - When set to `auto`, the software automatically detects when to keep the center line unchanged. *Default*: `auto` *Data_type*: enum, optional
`-no_connection`	Resizes the wire without auto via/wire creation and DRC checking. If `-no_connection` is not specified, the tool will reprocess the via and do basic DRC checking:  **Note:** Patch wires are resized without auto via/wire creation and DRC checking even if `-no_connection` is not specified. *Data_type*: bool, optional

`-offset` *float_value*	Specifies, in microns, the distance to which the wires or pins should be stretched or shrunk. *Data_type*: float, optional
`-polygon_edge_end {x y}`	Specifies the end point for a polygon edge {x y}. This option is used only to resize a polygon. *Data_type*: point, optional
`-polygon_edge_start {x y}`	Specifies the start point for a polygon edge {x y}. This option is used only to resize a polygon. *Data_type*: point, optional
`-side {high low}`	Specifies the side from which the wire or pin is to be stretched or shrunk: • `high` - Stretches or shrinks wires or pins from the top or the right. • `low` - Stretches or shrinks the wires or pins from the bottom or the left. *Data_type*: enum, optional
`-to` *float_value*	Specifies, in microns, the new height for selected horizontal wires or the new width for selected vertical wires. *Data_type*: float, optional

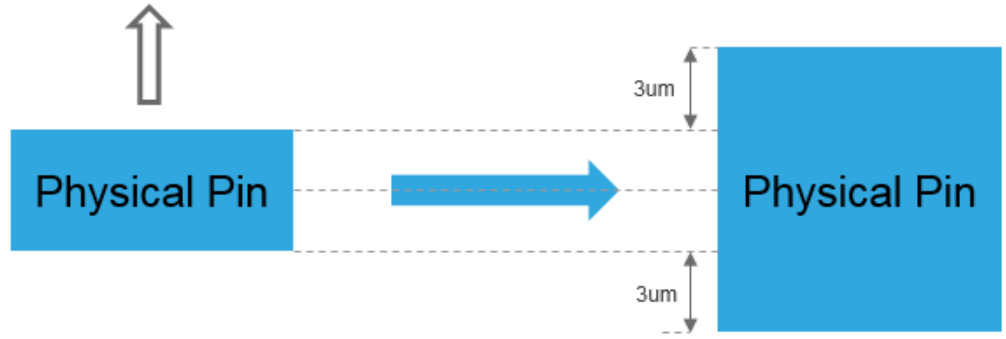
Examples

- The following command stretches the selected vertical wire from the top by 8.5 µm.
  ```
  edit_resize_routes -direction y -offset 8.5 -side high
  ```
- The following command stretches the selected vertical wire from the bottom by 0.511 µm.
  ```
  edit_resize_routes -direction y -offset -0.511 -side low
  ```
- The following command shrinks the selected vertical wire from the bottom by 0.619 µm.
  ```
  edit_resize_routes -direction y -offset 0.619 -side low
  ```
- The following command stretches a horizontal wire from the right by 1.27 µm.
  ```
  edit_resize_routes -direction x -offset 1.27 -side high
  ```
- The following command shrinks a horizontal wire from the left by 0.443 µm.
  ```
  edit_resize_routes -direction x -offset 0.443 -side low
  ```
- The following command stretches a horizontal wire from the left by 0.86 µm.
  ```
  edit_resize_routes -direction x -offset -0.86 -side low
  ```
- The following command increases the width of the selected horizontal special wire by 0.01 µm:
  ```
  edit_resize_routes -direction y -offset 0.01 -side high
  ```
- The following command changes the width of the selected horizontal regular wire to 0.48 µm:
  ```
  edit_resize_routes -to 0.48 -direction y -side high
  ```

If 0.48 is not defined in the NDR, the tool snaps the width to the closest value defined in the NDR.

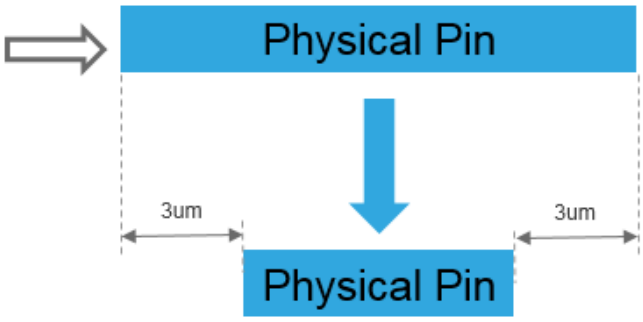
- The following command increases the vertical width of the selected physical pin by 3 µm each from both top and bottom, keeping the center line unchanged:

```
edit_resize_routes -direction y -offset -3 -side high -no_connection -keep_center_line 1
```



- The following command reduces the horizontal width of the selected physical pin by 3 µm each from both left and right, keeping the center line unchanged:

```
edit_resize_routes -direction x -offset 3 -side low -no_connection -keep_center_line 1
```



Related Information

- edit_add_route_point
- edit_adjust_routes
- edit_cut_route
- edit_duplicate_routes
- edit_end_route_point
- edit_merge_routes
- edit_move_routes
- edit_rotate
- edit_split_routes
- edit_trim_routes
- delete_routes
- gui_redo
- gui_undo
- select_routes
- set_db
- Attributes in the edit_wire category
- Editing Wires chapter in the *User Guide*
- Edit Menu chapter in the *Menu Reference*
  - *Wire* section

edit_rotate

```
edit_rotate
[-help]
[-mode {group | single}]
[-rotate_point {x y}]
{-flip {mx my} | -rotate {r90 r180 r270}}
```

Flips or rotates selected wires and vias. Use this command for more flexible custom routing and fixing DRC violations using existing wires. Using the `editRotate` command, you can:

- Flip selected wires or vias on the X or Y axis (`MX` or `MY`)
- Rotate the selected wires or vias by 90, 180, or 270 degrees (`R90`, `R180`, or `R270`).

The selected object is rotated or flipped based on current orientation, and the lower-left coordinate of the wire/via remains the same after rotation or flip.

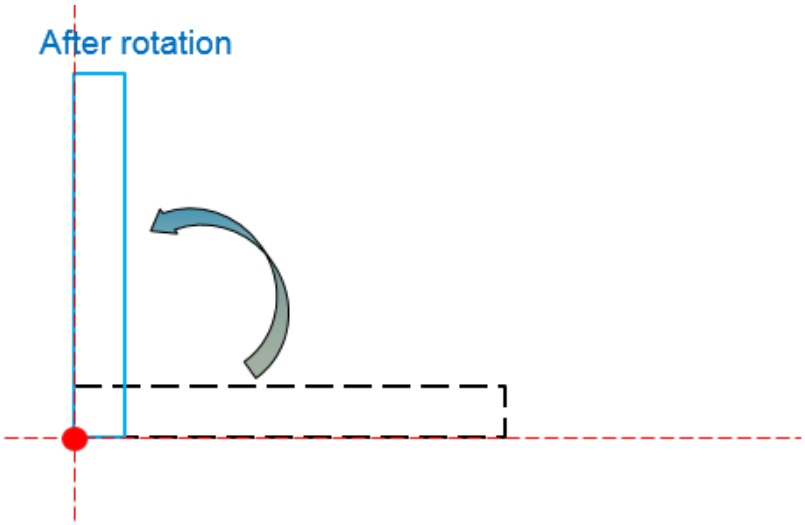
You can also flip or rotate multiple objects simultaneously as one group.

Parameters

`-help`	Prints the usage of the command. For a detailed description of the command, use the `man` command: `man editRotate`
`-flip {mx my}`	Flips a wire or via. If you specify `mx`, the selected object is flipped on the X axis. If you specify my, the object is flipped on the Y axis. *Data_type*: enum, optional
`-mode {group single}`	Rotates multiple selected wires and vias at the same time using the specified mode: • `group` - Flips or rotates the selected objects as a group. This is the default mode when multiple objects are selected. In this mode, the tool flips or rotates the bounding box which encloses all the selected objects. • `single` - Flips or rotates the selected objects individually. *Data_type*: enum, optional
`-rotate {r90 r180 r270}`	Rotates wires or vias by 90, 180, or 270 degrees, as specified. *Data_type*: enum, optional
`-rotate_point {x y}`	Specifies the point around which the wire should be rotated. If this option is not specified, the selected wires/vias are rotated around the lower-left corner of bounding box of wires/vias. *Data_type*: point, optional

Examples

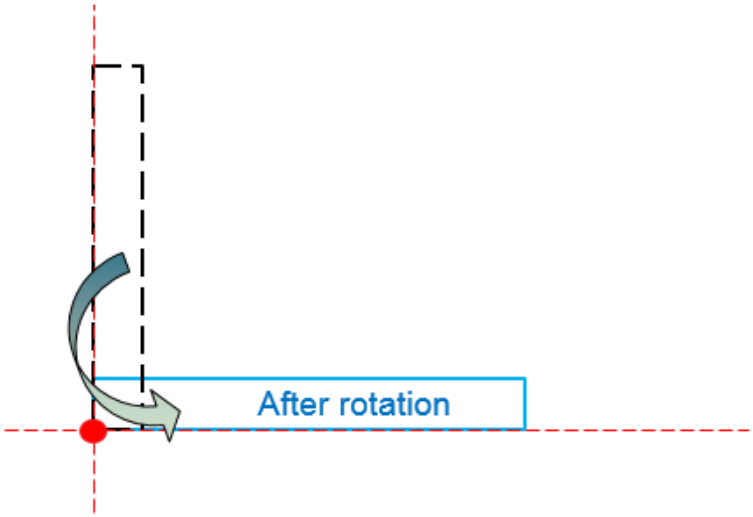
- The following command rotates the selected wire by 90 degrees:
  ```
  edit_rotate -rotate r90
  ```



- The following command rotates the selected object, which originally has `r90` as its original orientation by 90 degrees:

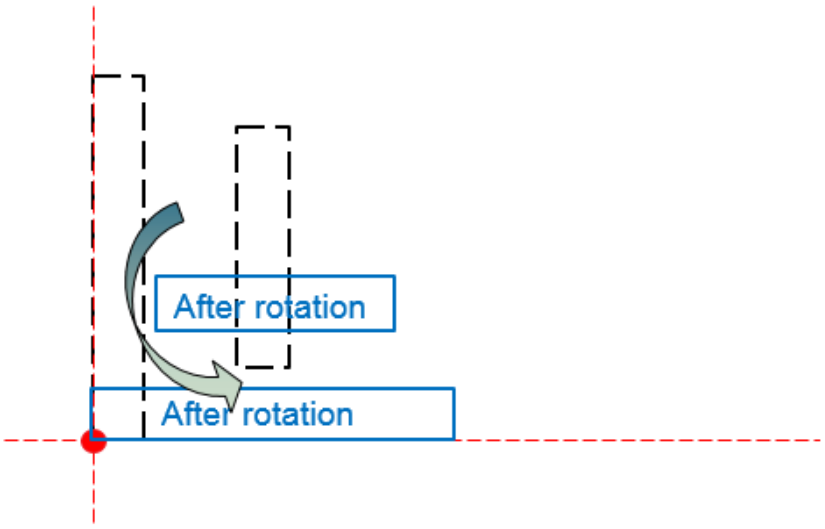
  ```
  edit_rotate -rotate r90
  ```

  The final orientation of the object will be `r180` as shown below:



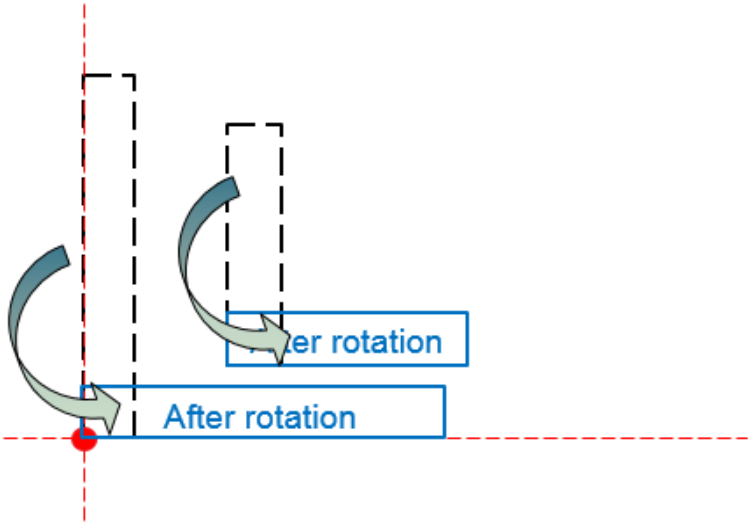
- The following command rotates the bounding box of the selected wires by 90 degrees to rotate the wires as one group:

  ```
  edit_rotate -rotate r90 -mode group
  ```



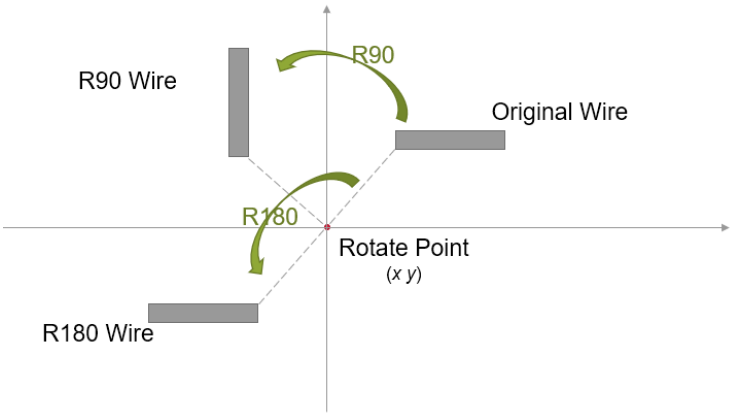
- The following command rotates the selected wires individually by 90 degrees:

```
edit_rotate -rotate r90 -mode single
```



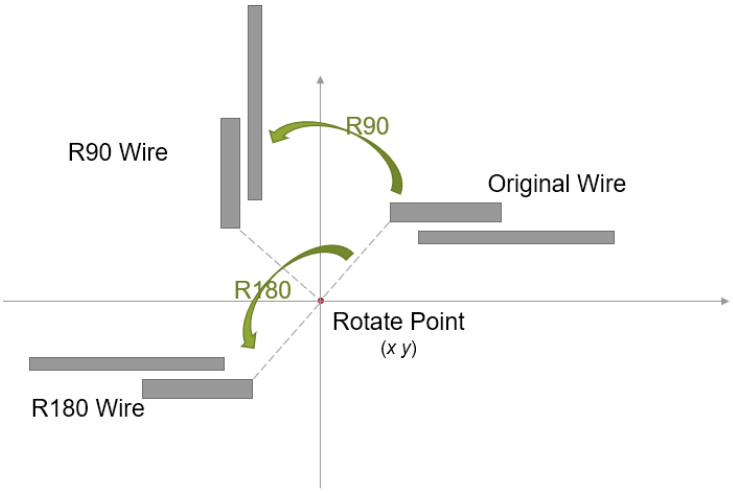
- The following commands rotate the selected wire by 90 and 180 degrees, respectively, around the specified rotation point, *x y*:

```
edit_rotate -rotate r90 -rotate_point x y
edit_rotate -rotate r180 -rotate_point x y
```



- The following commands rotate the selected group of wires by 90 and 180 degrees, respectively, around the specified rotation point, *x y*:

```
edit_rotate -rotate r90 -mode group -rotate_point x y
edit_rotate  -rotate r180 -mode group -rotate_point x y
```



Related Information

- `edit_add_route_point`
- `edit_adjust_routes`
- `edit_cut_route`
- `edit_duplicate_routes`
- `edit_end_route_point`
- `edit_merge_routes`
- `edit_move_routes`
- `edit_resize_routes`
- `edit_split_routes`
- `edit_trim_routes`
- `delete_routes`
- `gui_redo`
- `gui_undo`
- `select_routes`
- `set_db`
- Attributes in the `edit_wire` category
- Editing Wires chapter in the *User Guide*
- Edit Menu chapter in the *Menu Reference*
  - *Wire* section
- Floorplan Menu chapter in the *Menu Reference*
  - *Flip/Rotate* section

edit_split_routes

```
edit_split_routes [-help]
```

Splits selected special wires at connection points with orthogonal wires. This command can also be used to split selected bus guides.

Note: Use the `select_routes` command to select the wires. Use the `select_obj` command to select bus guides or bus guide segments

Parameters

`-help`	Prints the usage of the command. For a detailed description of the command, use the `man` command: `man edit_split_routes`

Related Information

- edit_add_route_point
- edit_adjust_routes
- edit_cut_route
- edit_duplicate_routes
- edit_end_route_point
- edit_merge_routes
- edit_move_routes
- edit_resize_routes
- edit_rotate
- edit_trim_routes
- delete_routes
- gui_redo
- gui_undo
- select_obj
- select_routes
- set_db
- Attributes in the edit_wire category
- Editing Wires chapter in the *User Guide*
- Edit Menu chapter in the *Menu Reference*
  - *Wire* section

edit_split_wide_routes

```
edit_split_wide_routes [-help]
```

Splits wires that violate the MAXWIDTH value in the LEF LAYER (Routing) statement. Also splits diagonal wires whose width is greater than the MAXWIDTH value as they are drawn.

Parameters

-help	Prints the usage of the command. For a detailed description of the command, use the man command: `man edit_split_wide_routes`

Related Information

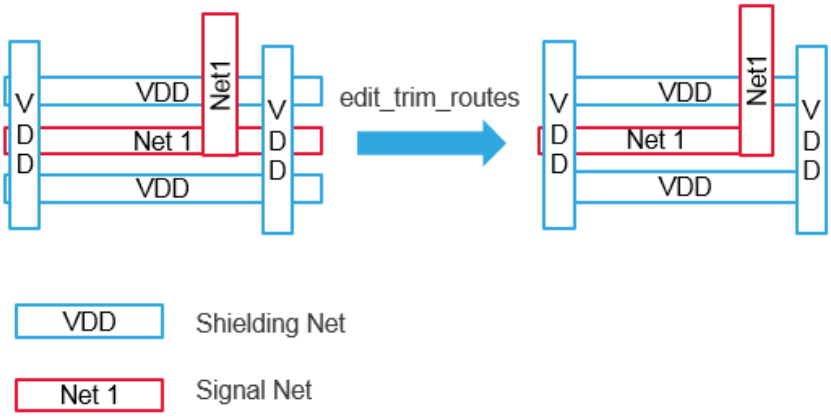
- edit_split_routes
- gui_undo
- gui_redo
- Attributes in the edit_wire category
- Editing Wires chapter in the *User Guide*
  - *Repairing Maximum Wire Width Violations*
- Flip Chip Methodologies chapter in the *User Guide*
  - *SPLIT Constraint*
- Edit Menu chapter in the *Menu Reference*
  - *Wire* section

edit_trim_routes

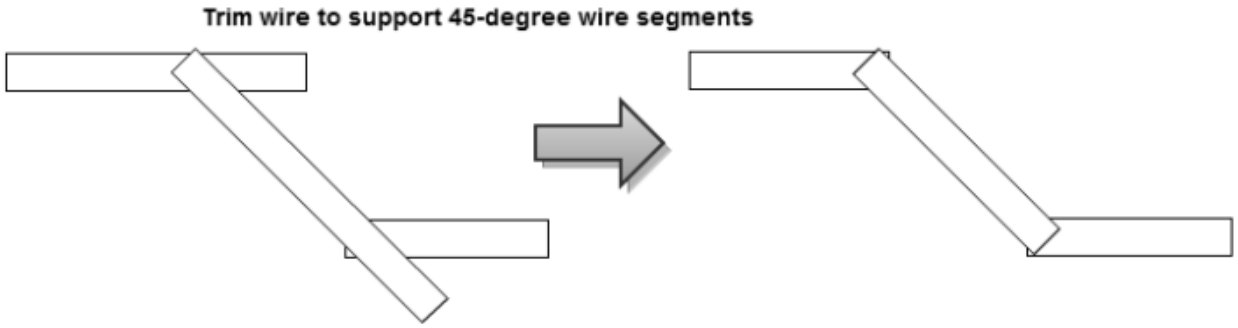
```
edit_trim_routes
[-help]
[-keep_floating_stubs]
[-status {cover fixed noshield routed shield}]
[-type {float}]
[-all  | -selected  | -nets netNames]
[-all  | -selected  | -layers layer_names]
[-all  | -selected  | -area {x1 y1 x2 y2}]
```

Removes or trims wires. Wires are trimmed or removed as follows:

- Trims floating stubs of wires back to the closest connection point.
- Removes wires that do not have electrical connections to other wires or pins (floating wires).
- Removes wires with only one connection to another wire or pin, along with connecting vias. Removing a wire with one connection might cause the wire to which the just-removed wire is connected to change from being a wire with two connections to a wire with one connection and thus it will also be removed. This process is repeated until no wires with one connection are left.
  Note: The `edit_trim_routes` command recognizes patch wires and detects the patch wire connection with the via. Therefore, it does not delete patch wires even if they are connected from only one side.
- Trims shield wires along with signal wires.

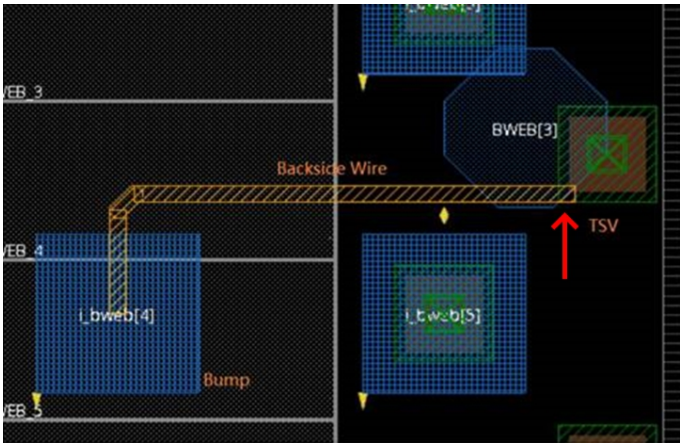


- Trims wires to support 45-degree wire segments.



Notes

- The `edit_trim_routes` command recognizes Through Silicon Via (TSV) connections and detects the backside wire connection with the TSV so it does not remove the backside routes connected to TSV.



- The `edit_trim_routes` command does not trim metal fill segments. To trim metal fill segments use the `trim_metal_fill` command.

Parameters

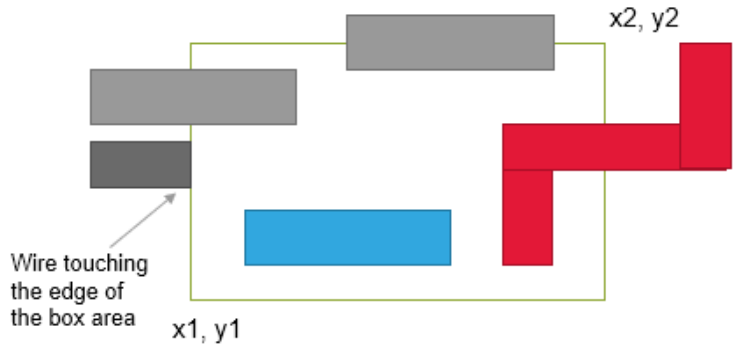
`-help`	Prints the usage of the command. For a detailed description of the command, use the `man` command: `man edit_trim_routes`
`-all`	Trims all special wires and floating stubs. *Data_type*: bool, optional

`-area {x1 y1 x2 y2}`	Specifies the coordinates for the box area from which to trim the wires. Here: • `x1` - Specifies the lower left x coordinate. • `y1` - Specifies the lower left y coordinate. • `x2` - Specifies the upper right x coordinate. • `y2` - Specifies the upper right y coordinate. Only the wires that overlap with the specified box area are trimmed. The wires that just touch the box area are not trimmed. *Data_type*: rect, optional
`-keep_floating_stubs`	Selects the floating wires but does not remove the floating stubs. Use this parameter if you want to assess the impact before actually trimming. *Data_type*: bool, optional
`-layers layer_names`	Specifies the layers from which to trim wires. If you are specifying multiple layers, enclose the layer names in braces. *Default*: If you do not specify this parameter, the command trims wires on all layers. *Data_type*: string, optional
`-nets net*`	Trims wires on the nets you specify. You can use wildcards to specify the nets. *Data_type*: (net)*, optional
`-selected`	Trims selected wires. Use the `select_routes` command to select the wires. This is the default value. *Data_type*: bool, optional
`-status {cover fixed noshield routed shield}`	
	Specifies the status of wires to trim. *Data_type*: enum, optional
`-type {float}`	Removes only wires that do not have electrical connections to other wires or pins (floating wires). The tool will not trim the wire with connection to other wires at one end if this option is specified. *Data_type*: enum, optional

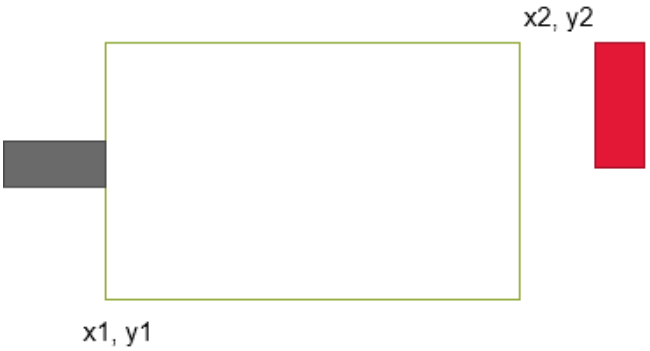
Examples

- The following command trims the wires overlapping the box area specified by the coordinates `{x1 y1 x2 y2}`:
  `edit_trim_routes -area {x1 y1 x2 y2}`

  Before



  After



- The following command trims the wires in the specified layers:
  `edit_trim_routes -layers {Metal1 Metal3}`

Related Information

- `edit_add_route_point`
- `edit_adjust_routes`
- `edit_cut_route`
- `edit_duplicate_routes`
- `edit_end_route_point`
- `edit_merge_routes`
- `edit_move_routes`
- `edit_resize_routes`
- `edit_rotate`
- `edit_split_routes`
- `delete_routes`
- `gui_redo`
- `gui_undo`
- `select_routes`
- `set_db`
- `trim_metal_fill`
- Attributes in the `edit_wire` category
- Editing Wires chapter in the *User Guide*
- Edit Menu chapter in the *Menu Reference*
  - *Wire* section

edit_update_route_layer

```
edit_update_route_layer
[-help]
[-layer_horizontal layer]
[-layer_vertical layer]
```

Changes the layer of selected wires and updates vias connecting to the wires that change layers.

Note: Use the `select_routes` command to select the wires.

Parameters

`-help`	Prints a brief description that includes type and default information for each `edit_update_route_layer` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man edit_update_route_layer`
`-layer_horizontal` *layer*	Specifies the new layer for the selected horizontal wires. *Data_type*: string, optional
`-layer_vertical` *layer*	Specifies the new layer for the selected vertical wires. *Data_type*: string, optional

Example

- The following command moves all selected horizontal wires to the *METAL3* layer and all selected vertical wires to the *METAL4* layer:
  ```
  edit_update_route_layer -layer_horizontal METAL3 -layer_vertical METAL4
  ```

Related Information

- `delete_routes`
- `edit_update_route_mask`
- `edit_update_route_net`
- `edit_update_route_rule`
- `edit_update_route_status`
- `edit_update_via`
- `gui_undo`
- `gui_redo`
- `select_routes`
- Attributes in the `edit_wire` category
- Editing Wires chapter in the *User Guide*
- Edit Menu chapter in the *Menu Reference*
    - *Wire* section

edit_update_route_mask

```
edit_update_route_mask
[-help]
{-to {0 1 2} | [[-top {0 1 2}] [-bottom {0 1 2}] [-cut {0 1 2}]] | -trim_metal {0 1 2}}
```

Changes the mask for the selected wires, vias, physical pins, or trim metal to the specified color.

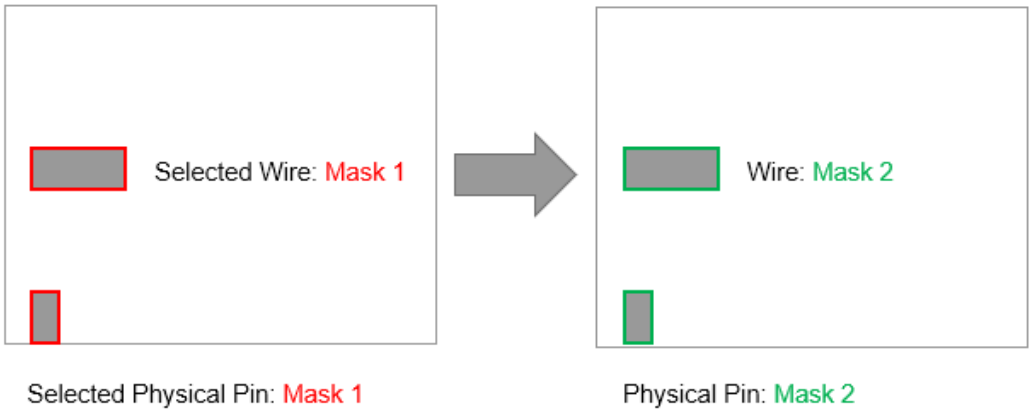
Parameters

`-help`	Prints a brief description that includes type and default information for each `edit_update_route_mask` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man edit_update_route_mask`
`-bottom {0 1 2}`	Changes the mask for the bottom layer of the selected vias to the specified color. *Data_type*: enum, optional
`-cut {0 1 2}`	Changes the mask for the cut layer of the selected vias to the specified color. *Data_type*: enum, optional
`-to {0 1 2}`	Changes the mask for the selected wires and physical pins to the specified color. *Data_type*: enum, optional
`-top {0 1 2}`	Changes the mask for the bottom layer of the selected vias to the specified color. *Data_type*: enum, optional
`-trim_metal {0 1 2}`	Changes the mask for the trim metal of the selected patch wires to the specified color. If the trim metal being edited completely covers another trim metal, the mask of the covered trim metal is also changed simultaneously. *Data_type*: enum, optional

Examples

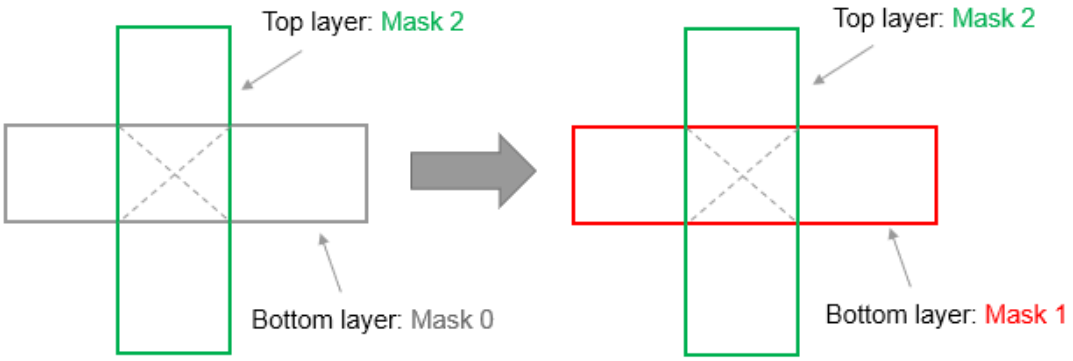
- The following command changes the mask for the selected wire and physical pin to `2`:

  ```
  edit_update_route_mask -to {2}
  ```



- The following command changes the mask for the bottom layer of the selected via to `1`:

  ```
  edit_update_route_mask -bottom {1}
  ```

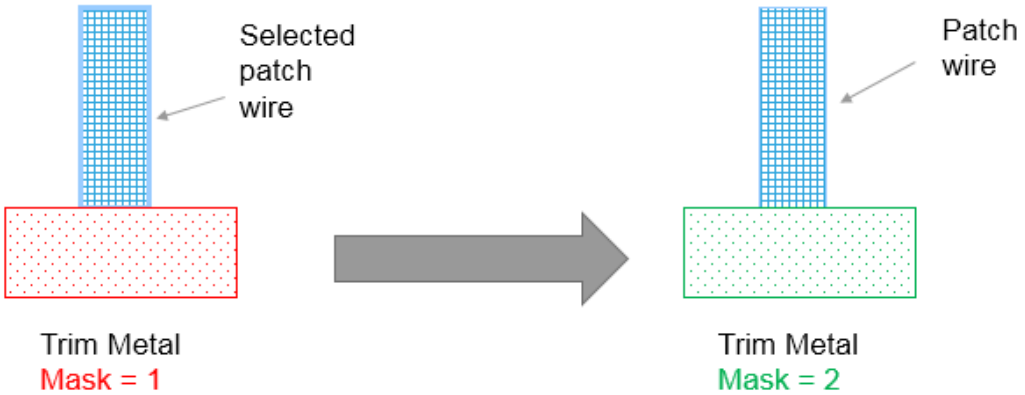


- The following command changes the mask for the trim metal of the selected patch wires to `2`:

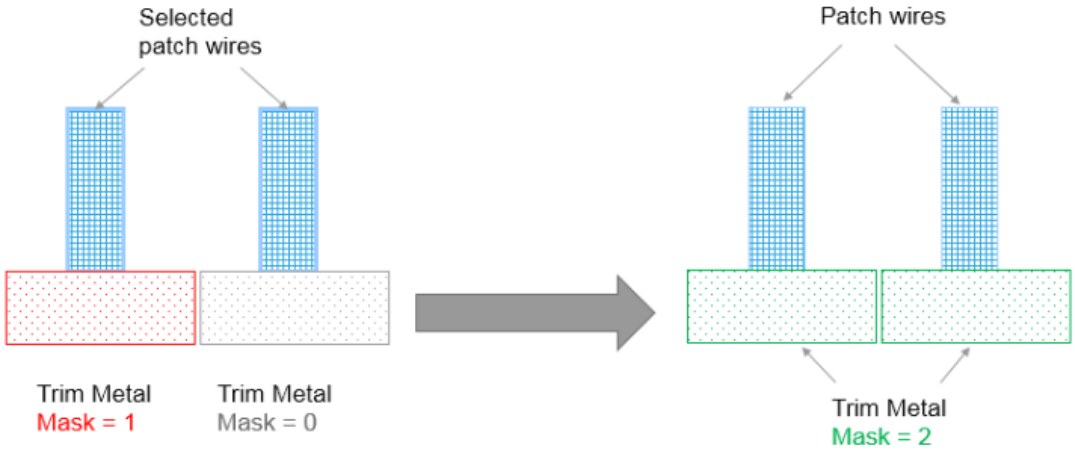
  ```
  edit_update_route_mask -trim_metal {2}
  ```

  Given below are the results of running this command in various situations:

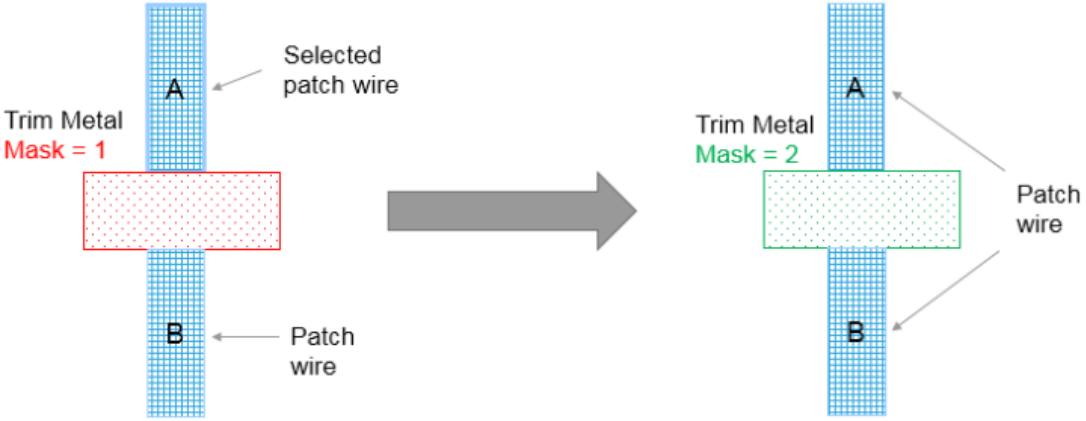
  - If a single patch wire was selected before running the command:



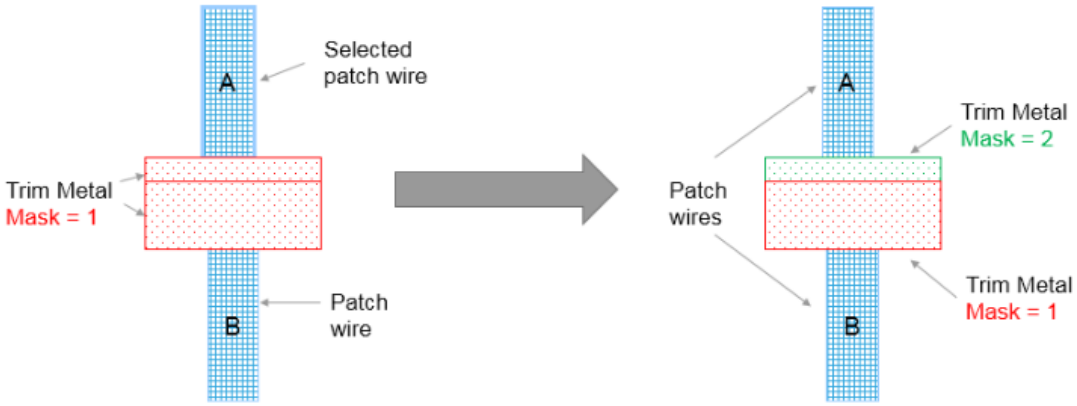
  - If multiple patch wires were selected before running the command:



- If the trim metal for the selected patch wire (Patch Wire A) completely overlaps the trim metal for another patch wire that is not selected (Patch Wire B):



- If the trim metal for the selected patch wire (Patch Wire A) partially overlaps the trim metal for another patch wire that is not selected (Patch Wire B):



Related Information

- `delete_routes`
- `edit_add_trim_metal`
- `edit_update_route_layer`
- `edit_update_route_net`
- `edit_update_route_rule`
- `edit_update_route_status`
- `edit_update_via`
- `gui_undo`
- `gui_redo`
- `select_routes`
- Attributes in the `edit_wire` category
- Editing Wires chapter in the *User Guide*
- Edit Menu chapter in the *Menu Reference*
  - *Wire* section

edit_update_route_net

```
edit_update_route_net
[-help]
-to netName
```

Changes the net associated with selected physical pins, wires, and vias.

You can also use `edit_update_route_net` to change the net name of the patch wire (pWire) generated by NanoRoute.

Parameters

`-help`	Prints a brief description that includes type and default information for each `edit_update_route_net` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man edit_update_route_net`
`-to`	Specifies the name of the net to which to assign the selected physical pins, wires and vias. *Data_type*: (net), required

Example

- The following set of commands selects the physical pins, wires, and vias that belong to net `VDD1` and reassigns them to net `VDD2`:

  ```
  select_routes -nets VDD1
  edit_update_route_net -to VDD2
  ```

Related Information

- `delete_routes`
- `edit_add_trim_metal`
- `edit_update_route_layer`
- `edit_update_route_mask`
- `edit_update_route_rule`
- `edit_update_route_status`
- `edit_update_via`
- `gui_undo`
- `gui_redo`
- `select_routes`
- Attributes in the `edit_wire` category
- Editing Wires chapter in the *User Guide*
- Edit Menu chapter in the *Menu Reference*
  - *Wire* section

edit_update_route_rule

```
edit_update_route_rule
[-help]
[-from old_rule]
-to new_rule
```

Changes the LEF rule associated with selected wire segments from the *old_rule* to the *new_rule*. You must select the wires for which the rule needs to be changed before running edit_update_route_rule. You can select wires either using the GUI or by using the select_routes command.

Note: This command does not change the routing rule for the net. If the net is re-routed by NanoRoute, these changes will be lost. To change the routing rule, use set_route_attributes -nets -route_rule. For more information, see set_route_attributes.

Parameters

-help	Prints a brief description that includes type and default information for each edit_update_route_rule parameter. For a detailed description of the command and all of its parameters, use the man command: man edit_update_route_rule
-from *old_rule*	Specifies the qualifying rule. Only wire segments whose associated rule matches *old_rule* are changed. If you do not specify this parameter, the tool will change the rule of all selected wires. *Data_type*: string, optional
-to *new_rule*	Specifies the destination rule. Wire segments originally associated with *old_rule* are associated with *new_rule* after this command runs. Specify one of the following: • A nondefault rule defined in the LEF LAYER (Routing) section • default (applies the default routing rule) *Data_type*: string, required

Example

- The following command changes the wire segments that are associated with rule SP to the default rule. If the wire width or extension of SP is different from its counterpart in the default rule, the wire geometry is changed accordingly.
  ```
  edit_update_route_rule -from SP -to default
  ```

Related Information

- `delete_routes`
- `edit_update_route_layer`
- `edit_update_route_mask`
- `edit_update_route_net`
- `edit_update_route_status`
- `edit_update_via`
- `gui_undo`
- `gui_redo`
- `select_routes`
- `set_route_attributes`
- Attributes in the `edit_wire` category
- Editing Wires chapter in the *User Guide*
- Edit Menu chapter in the *Menu Reference*
    - *Wire* section

edit_update_route_status

```
edit_update_route_status
[-help]
[shield_net_name]
[-nets <net>+]
-to {routed fixed cover shield noshield}
```

Changes the DEF status of selected wires or vias to the specified status. Use this command to change the net status of the selected net. If `-nets` is specified with `edit_update_route_status`, the tool changes the status of all the objects belonging to the specified nets to the status specified with the `-to` parameter.

Note: Before using this command to change the status of a wire, you must select the wire by either clicking it or using the `select_routes` command.

Parameters

`-help`	Prints a brief description that includes type and default information for each `edit_update_route_status` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man edit_update_route_status`
`shield_net_name`	Specifies the name of the net to be shielded by the selected wire. Use this parameter if you want to change the selected wire to a `shield` for the specified net. **Note**: You can use this parameter only if you have specified `-to shield`. *Data_type*: string, optional
`-nets <net>+`	Specifies the names of the nets for which the object status is to be changed. If `-nets` is specified with `edit_update_route_status`, the tool changes the status of all the objects (regular wires, special wires, regular vias, and special vias) belonging to the specified nets to the status specified with the `-to` parameter. *Data_type*: (net), optional
`-to {routed fixed cover shield noshield}`	
	Specifies the DEF status of the selected wires or vias after you issue this command. **Note:** You cannot change a special wire to `noshield` status or change a signal wire to `shield` status. *Data_type*: enum, required

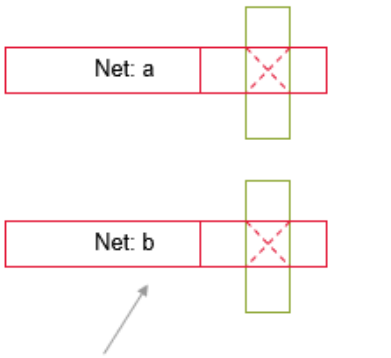
Examples

- The following commands select and then freeze the wires and vias on the `METAL1` and `METAL2` layers by

changing their status to `cover`. Innovus does not delete wires or vias with status `cover`:

```
select_routes -layer {METAL1 METAL2}
edit_update_route_status -to cover
```

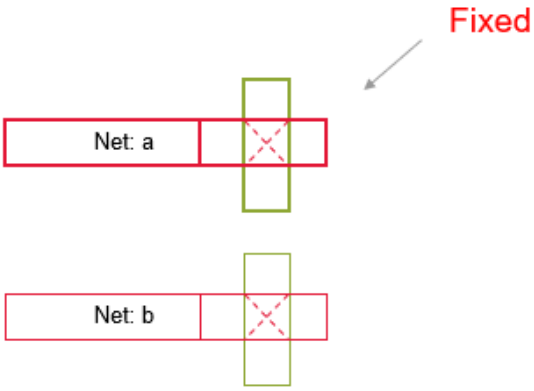
- Consider the nets depicted below:



The wire/via under net b is **selected**

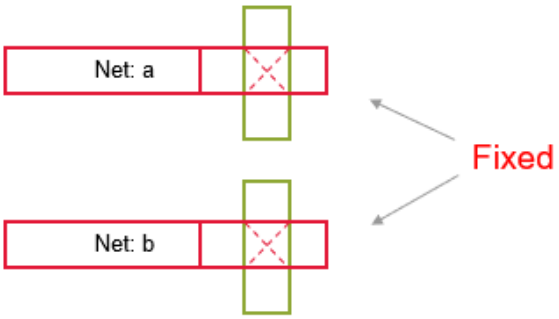
  - The following command changes the status of only the wire and via under net `a` to `fixed`. The status of the wire and via under net `b` is not changed even though net `b` is selected:

    ```
    edit_update_route_status -nets a -to fixed
    ```



  - The following command changes the status of the wires and vias under both nets `a` and `b` to `fixed`:

    ```
    edit_update_route_status -nets {a b} -to fixed
    ```



Related Information

- `delete_routes`
- `edit_update_route_layer`
- `edit_update_route_mask`
- `edit_update_route_net`
- `edit_update_route_rule`
- `edit_update_via`
- `gui_undo`
- `gui_redo`
- `select_routes`
- Attributes in the `edit_wire` category
- Editing Wires chapter in the *User Guide*
- Edit Menu chapter in the *Menu Reference*
    - *Wire* section

edit_update_via

```
edit_update_via
[-help]
[-net net_name]
[-signal_only {0 | 1}]
{[-from string [-area {x1 y1 x2 y2} | -at {x y} ]] | [-selected]}
{-to string | [-via_columns integer -via_rows integer [-cut_class class_name]]}
```

Replaces the selected vias or the vias listed in `-from` with the specified vias or with vias having the specified cut class and pattern.

While modifying a via with `edit_update_via`, the tool behavior varies depending on your `edit_wire_drc_on` and `edit_wire_keep_via` attribute settings:

- `set_db edit_wire_drc_on false`
  With this setting, the tool does not check for Design Rule Check (DRC) violations when you modify a via using the `edit_update_via` command.
- `set_db edit_wire_drc_on true`
  `set_db edit_wire_keep_via false`
  With these settings, the tool checks for DRC violations and allows you to replace the via even if the new via is generating violations. The DRC violations are displayed on the layout.
- `set_db edit_wire_drc_on true`
  `set_db edit_wire_keep_via true`
  With these settings, the tool checks for DRC violations and retains the original via if the new one is generating violations.

The behavior defined above applies for all via modifications with `edit_update_via`, irrespective of whether you use the `-to string` option or the `-via_columns integer -via_rows integer` options.

Note that the `edit_wire_drc_on true` setting does not check cut spacing, enclosure spacing, or other geometry of the via cell itself.

Parameters

`-help`	Prints a brief description that includes type and default information for each `edit_update_via` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man edit_update_via`
`-area {x1 y1 x2 y2}`	Specifies an area in which to replace vias. If a via indicated by the `-from` parameter is in this area, replaces the via with the via indicated by the `-to` parameter. *Data_type*: rect, required

`-at {x y}`	Specifies the coordinates for the origin of a via to replace. The via at this location is replaced by the via indicated by the `-to` parameter. *Data_type*: point, required
`-cut_class class_name`	Specifies the cut class of the replacement vias. *Data_type*: string, optional
`-from string`	Specifies a via or list of vias to replace. Enclose the via names on the list in curly braces and separate each name with a space. Vias whose masters are not specified in `string` are not changed. **Notes:** • You must specify the `-from` or the `-selected` parameter. If you do not specify at least one of these parameters, the software does not change any vias. • The number of via names on the `-from` list must be equal to the number of via names on the `-to` list. In addition, the corresponding pair of vias in each list must be on the same layer. *Data_type*: string, required
`-net net_name`	Specifies the net name on which to change vias. *Data_type*: (net), optional
`-selected`	Replace vias for selected ones only. **Note:** You must specify either the `-from` or the `-selected` parameter. If you do not specify at least one of these parameters, the software does not change any vias. *Data_type*: bool, optional
`-signal_only {1 | 0}`	Specifies whether signal vias only or signal and special vias are changed. Specify `1` to change signal via only. Specify `0` to change signal and special vias. *Default:* `0` *Data_type*: enum, optional
`-to string`	Specifies a via or a list of vias to replace the vias indicated by the `-from` parameter. If you specify more than one via, enclose the via list in curly braces and separate each via name with a space. **Note:** The number of via names on the list must be the equal to the number of via names on the list for the `-from` parameter. In addition, the corresponding pair of vias in each list must be on the same layer. *Data_type*: string, required
`-via_columns integer`	Specifies the number of columns in the replacement vias. *Data_type*: int, optional

`-via_rows` *integer*	Specifies the number of rows in the replacement vias. *Data_type*: int, optional

Example

- The following command changes all `VIA_XX` vias of the net `VDD1` located within a specified region to `VIA_YY` vias:
  ```
  edit_update_via -net VDD1 -area {x1 y1 x2 y2} -from VIA_XX -to VIA_YY
  ```
- The following command changes the selected vias with vias having a cut pattern of 3 by 3:
  ```
  edit_update_via -selected -via_columns 3 -via_rows 3
  ```

Related Information

- `delete_routes`
- `edit_update_route_layer`
- `edit_update_route_mask`
- `edit_update_route_net`
- `edit_update_route_rule`
- `edit_update_route_status`
- `gui_undo`
- `gui_redo`
- `select_routes`
- Attributes in the `edit_wire` category
- Editing Wires chapter in the *User Guide*
- Edit Menu chapter in the *Menu Reference*
  - *Wire* section

edit_wire_via Category Attributes

The root attributes in the `edit_wire` category are:

- edit_wire_via_allow_geometry_drc
- edit_wire_via_auto_snap
- edit_wire_via_auto_update
- edit_wire_via_cell_name
- edit_wire_via_create_by
- edit_wire_via_cut_layer
- edit_wire_via_snap_honor_color
- edit_wire_via_type

To set the value of any root attribute, use the `set_db` command:

```
set_db attribute_name value
```

To get the current value of any root attribute, use the `get_db` command:

```
get_db attribute_name
```

To get the current value of all attributes in the `edit_wire` category, use the following command:

```
get_db -category edit_wire
```

The following set of commands add a via with the same geometry as the cell specified by the `edit_wire_via_cell_name` attribute. The origin of the via is at coordinates `550`, `675`:

```
set_db edit_wire_via_create_by viacell

edit_create_via 550 675
```

For more help on editing wires, see the Editing Wires chapter in the *User Guide.*

edit_wire_via_allow_geometry_drc

```
edit_wire_via_allow_geometry_drc {true|false}
```

Default: `true`

Data_type: bool, read/write

Checks geometry DRC.

Related Commands

To review global controls that affect wire editing, see `edit_wire Category Attributes` in the *Text Command Reference*. Other related commands are:

- `delete_routes`

- edit_create_via
- edit_move_routes
- edit_resize_routes
- edit_update_route_layer
- edit_update_route_net
- edit_update_route_rule
- edit_update_route_status
- edit_update_via
- select_routes
- edit_trim_routes

edit_wire_via_auto_snap

```
edit_wire_via_auto_snap {true|false}
```

Default: `true`

Data_type: bool, read/write

Controls whether to snap vias to the intersection of tracks or to the manufacturing grid.

- Specify `true` to snap the via to the center of the intersection of tracks.
- Specify `false` to keep the via in the location where it is added or moved, snapping only to the manufacturing grid.

Related Commands

To review global controls that affect wire editing, see `edit_wire Category Attributes` in the *Text Command Reference*. Other related commands are:

- delete_routes
- edit_create_via
- edit_resize_routes
- edit_move_routes
- edit_update_route_layer
- edit_update_route_net
- edit_update_route_rule
- edit_update_route_status

- edit_update_via
- select_routes
- edit_trim_routes

edit_wire_via_auto_update

```
edit_wire_via_auto_update {true|false}
```

Default: `true`

Data_type: bool, read/write

Determines whether existing vias are replaced with new ones in case of overlap.

- Specify `true` to replace existing vias with the new ones added with `edit_create_via` in case of any overlap.
- Specify `false` if you want to retain existing vias even if there is some overlap with the new vias being added.

Related Commands

To review global controls that affect wire editing, see `edit_wire Category Attributes` in the *Text Command Reference*. Other related commands are:

- delete_routes
- edit_create_via
- edit_move_routes
- edit_resize_routes
- edit_update_route_layer
- edit_update_route_net
- edit_update_route_rule
- edit_update_route_status
- edit_update_via
- select_routes
- edit_trim_routes

edit_wire_via_cell_name

```
edit_wire_via_cell_name cellname
```

Default: `""`

Data_type: string, read/write

Specifies the name of the via cell from the LEF or DEF file.

Related Commands

To review global controls that affect wire editing, see `edit_wire Category Attributes` in the *Text Command Reference*. Other related commands are:

- `delete_routes`
- `edit_create_via`
- `edit_move_routes`
- `edit_resize_routes`
- `edit_update_route_layer`
- `edit_update_route_net`
- `edit_update_route_rule`
- `edit_update_route_status`
- `edit_update_via`
- `select_routes`
- `edit_trim_routes`

edit_wire_via_create_by

`edit_wire_via_create_by {viacell|parameters}`

Default: `parameters`

Data_type: enum {viacell parameters}, read/write

Controls whether the tool creates vias based on the `edit_wire_via_cell_name` attribute or the cut class name.

- Specify `viacell` to create the via with the same geometry as the cell specified by the `edit_wire_via_cell_name` attribute.
- Specify `parameters` to create the via cell based on the cut class parameters.

Related Commands

To review global controls that affect wire editing, see `edit_wire Category Attributes` in the *Text Command Reference*. Other related commands are:

- `delete_routes`
- `edit_create_via`

- edit_move_routes
- edit_resize_routes
- edit_update_route_layer
- edit_update_route_net
- edit_update_route_rule
- edit_update_route_status
- edit_update_via
- select_routes
- edit_trim_routes

edit_wire_via_cut_layer

`edit_wire_via_cut_layer` *cellname*

Default: `V12`

Data_type: string, read/write

Specifies the via layer name for the via to be created or modified.

Related Commands

To review global controls that affect wire editing, see `edit_wire Category Attributes` in the *Text Command Reference*. Other related commands are:

- delete_routes
- edit_create_via
- edit_move_routes
- edit_resize_routes
- edit_update_route_layer
- edit_update_route_net
- edit_update_route_rule
- edit_update_route_status
- edit_update_via
- select_routes
- edit_trim_routes

edit_wire_via_snap_honor_color

`edit_wire_via_snap_honor_color {true|false}`

Default: `true`

Data_type: bool, read/write

Specifies whether or not to snap via to the track with the same color automatically.

Related Commands

To review global controls that affect wire editing, see `edit_wire Category Attributes` in the *Text Command Reference*. Other related commands are:

- `delete_routes`
- `edit_create_via`
- `edit_move_routes`
- `edit_resize_routes`
- `edit_update_route_layer`
- `edit_update_route_net`
- `edit_update_route_rule`
- `edit_update_route_status`
- `edit_update_via`
- `select_routes`
- `edit_trim_routes`

edit_wire_via_type

`edit_wire_via_type {auto|regular|special}`

Default: `auto`

Data_type: enum {auto regular special}, read/write

Specifies the type of the via to be created:

- `auto` - Assigns the correct via type based on the type of connected wire.
- `regular` - Creates regular vias
- `special` - Creates special vias

For example, if you specify the type as `regular` by using the following command and then run the `edit_create_via` command, the tool will create a regular via even if there are PG shapes on other layers

overlapping the area:

```
set_db edit_wire_via_type regular
```

Related Commands

- delete_routes
- edit_add_route_point
- edit_create_via
- edit_merge_routes
- edit_move_routes
- edit_resize_routes
- edit_trim_routes
- edit_update_route_layer
- edit_update_route_net
- edit_update_route_rule
- edit_update_route_status
- edit_update_via
- select_routes

edit_wire Category Attributes

The root attributes in the `edit_wire` category are:

- edit_wire_align
- edit_wire_allow_45_degree
- edit_wire_arrow_incremental
- edit_wire_assign_multi_pattern_color
- edit_wire_auto_distribute_bus
- edit_wire_auto_split_bus
- edit_wire_bus_honor_start_parameters
- edit_wire_bus_honor_width_setting
- edit_wire_bus_total_width_horizontal
- edit_wire_bus_total_width_vertical
- edit_wire_change_order_at_turn
- edit_wire_check_design_boundary
- edit_wire_close_polygons
- edit_wire_color_align_with_track
- edit_wire_connect_pin
- edit_wire_connect_with_specified_layer
- edit_wire_create_crossover_vias
- edit_wire_create_is_edit_flag
- edit_wire_create_via_on_pin
- edit_wire_cut_class
- edit_wire_cut_wire_overlap
- edit_wire_debug_file
- edit_wire_delete_tsv
- edit_wire_delete_pin_with_wire
- edit_wire_delete_wire_via_through_layers
- edit_wire_display_wire_length_with_cursor
- edit_wire_draw_shield
- edit_wire_draw_with_group_centerline
- edit_wire_drawing_wire

- edit_wire_drc_aware_cross_metal
- edit_wire_drc_on
- edit_wire_drc_use_non_default_spacing
- edit_wire_extend_wires
- edit_wire_final_check_with_verify
- edit_wire_ignore_drc
- edit_wire_jog_connect_layer
- edit_wire_keep_floating_via
- edit_wire_keep_status
- edit_wire_keep_via
- edit_wire_lateral_movement_range
- edit_wire_layer
- edit_wire_layer_horizontal
- edit_wire_layer_max
- edit_wire_layer_min
- edit_wire_layer_vertical
- edit_wire_look_down_layers
- edit_wire_look_up_layers
- edit_wire_max_pointer_number
- edit_wire_mirror_bus_route
- edit_wire_nets
- edit_wire_no_merge_special_wire
- edit_wire_only_show_edit_layer
- edit_wire_orthogonal_connection_only
- edit_wire_outer_shield_spacing
- edit_wire_outer_shield_width
- edit_wire_override
- edit_wire_partial_overlap_threshold
- edit_wire_pull_back_distance
- edit_wire_reshape
- edit_wire_return_obj_pointer
- edit_wire_rule

- edit_wire_search_route_rule_vias_only
- edit_wire_shape
- edit_wire_shield
- edit_wire_shield_look_down_layers
- edit_wire_shield_look_up_layers
- edit_wire_shield_shape
- edit_wire_shielding_nets
- edit_wire_show_drc_info_for_edit_shape
- edit_wire_sibling_look_down_layers
- edit_wire_sibling_look_up_layers
- edit_wire_snap
- edit_wire_snap_align_to
- edit_wire_snap_bus_to_pin
- edit_wire_snap_end_to
- edit_wire_snap_objects_to_track
- edit_wire_snap_to
- edit_wire_snap_to_track_honor_color
- edit_wire_snap_trim_metal_to_trim_grid
- edit_wire_spacing
- edit_wire_spacing_horizontal
- edit_wire_spacing_vertical
- edit_wire_status
- edit_wire_stop_at_drc
- edit_wire_stretch_end
- edit_wire_stretch_with_intersection
- edit_wire_sub_class
- edit_wire_turn_at
- edit_wire_type
- edit_wire_unrestricted_regular_wire_width
- edit_wire_update_shield_net
- edit_wire_use_fix_via
- edit_wire_use_interleaving_wire_group

- edit_wire_use_wire_group
- edit_wire_use_wire_group_bits
- edit_wire_use_wire_group_reinforcement
- edit_wire_use_wire_group_reinforcement_group_via
- edit_wire_use_wire_group_reinforcement_spacing
- edit_wire_use_wire_group_reinforcement_width
- edit_wire_verbose
- edit_wire_via_columns
- edit_wire_via_exclude_spec
- edit_wire_via_override_spec
- edit_wire_via_rows
- edit_wire_via_scale_height
- edit_wire_via_scale_width
- edit_wire_via_snap_to_intersection
- edit_wire_width
- edit_wire_width_horizontal
- edit_wire_width_vertical
- edit_wire_wire_override_spec

To set the value of any root attribute, use the `set_db` command:

```
set_db attribute_name value
```

To get the current value of any root attribute, use the `get_db` command:

```
get_db attribute_name
```

To get the current value of all attributes in the `edit_wire` category, use the following command:

```
get_db -category edit_wire
```

For more help on editing wires, see the Editing Wires chapter in the *User Guide.*

edit_wire_align

```
edit_wire_align {true | false}
```

Default: `true`

Data_type: bool, read/write

Specifies whether or not to support alignment of objects during interactive edits using the GUI.

Related Commands

- delete_routes
- edit_add_route_point
- edit_merge_routes
- edit_move_routes
- edit_resize_routes
- edit_trim_routes
- edit_update_route_layer
- edit_update_route_net
- edit_update_route_rule
- edit_update_route_status
- edit_update_via
- select_routes

edit_wire_allow_45_degree

```
edit_wire_allow_45_degree {true | false}
```

Default: `false`

Data_type: bool, read/write

Specifies whether to allow the creation of a 45-degree wire.

Related Commands

- delete_routes
- edit_add_route_point
- edit_merge_routes
- edit_move_routes
- edit_resize_routes
- edit_trim_routes
- edit_update_route_layer
- edit_update_route_net
- edit_update_route_rule
- edit_update_route_status
- edit_update_via
- select_routes

edit_wire_arrow_incremental

`edit_wire_arrow_incremental` *value*

Default: `1.0`

Data_type: double, read/write

Specifies the step increments in microns to move wires with the arrow keys.

Related Commands

- delete_routes
- edit_add_route_point
- edit_merge_routes
- edit_move_routes
- edit_resize_routes
- edit_trim_routes
- edit_update_route_layer
- edit_update_route_net
- edit_update_route_rule
- edit_update_route_status
- edit_update_via
- select_routes

edit_wire_assign_multi_pattern_color

```
edit_wire_assign_multi_pattern_color {auto mask1 mask2 mask3}
```

Default: `auto`

Data_type: enum {auto mask1 mask2 mask3}, read/write

Specifies how to assign mask color on the double patterning technology (DPT) layer of the wire segment to be created. Select one of the following:

- `auto`: Mask color is assigned automatically.
- `mask1`: Assigns `mask1` color to the added wire segment.
- `mask2`: Assigns `mask2` color to the added wire segment.
- `mask3`: Assigns `mask3` color to the added wire segment.

Related Commands

- delete_routes
- edit_add_route_point
- edit_merge_routes
- edit_move_routes
- edit_resize_routes
- edit_trim_routes
- edit_update_route_layer
- edit_update_route_net
- edit_update_route_rule
- edit_update_route_status
- edit_update_via
- select_routes

edit_wire_auto_distribute_bus

```
edit_wire_auto_distribute_bus {true | false}
```

Default: `false`

Data_type: bool, read/write

Auto-distributes a group of wires, and sets the same spacing based on option `-bus_total_width_vertical` or `-bus_total_width_horizontal`.

If `-auto_distribute_bus` is set to `true`, the tool ignores the `-spacing_horizontal` and `-spacing_vertical` settings.

While drawing a bus, the tool auto-sets spacing as:
`wire spacing = (total bus width - total wire width)/ (wire number - 1)`

If the spacing is less than the minimum spacing, the tool auto-sets it to min spacing.

Related Commands

- delete_routes
- edit_add_route_point
- edit_merge_routes
- edit_move_routes
- edit_resize_routes
- edit_trim_routes
- edit_update_route_layer
- edit_update_route_net
- edit_update_route_rule
- edit_update_route_status
- edit_update_via
- select_routes

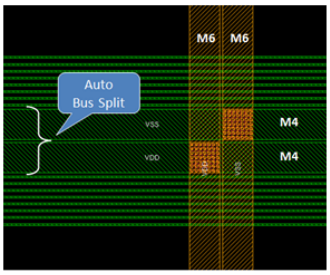
edit_wire_auto_split_bus

```
edit_wire_auto_split_bus {true | false}
```

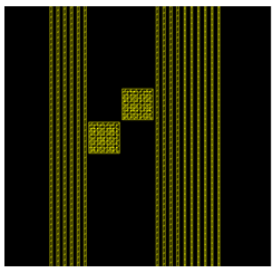
Default: `false`

Data_type: bool, read/write

Specifies whether or not Wire Editor should split buses automatically around power structures to avoid DRC violations.



Auto bus split around power trunk



Auto bus split around power vias

By default, there is no detour around via array and anchor wires of the bus being drawn always snap to the routing tracks.

`edit_wire_auto_split_bus` also supports busses of special wires with different widths.

Related Commands

- `delete_routes`
- `edit_add_route_point`
- `edit_merge_routes`
- `edit_move_routes`
- `edit_resize_routes`
- `edit_trim_routes`
- `edit_update_route_layer`
- `edit_update_route_net`
- `edit_update_route_rule`
- `edit_update_route_status`
- `edit_update_via`
- `select_routes`

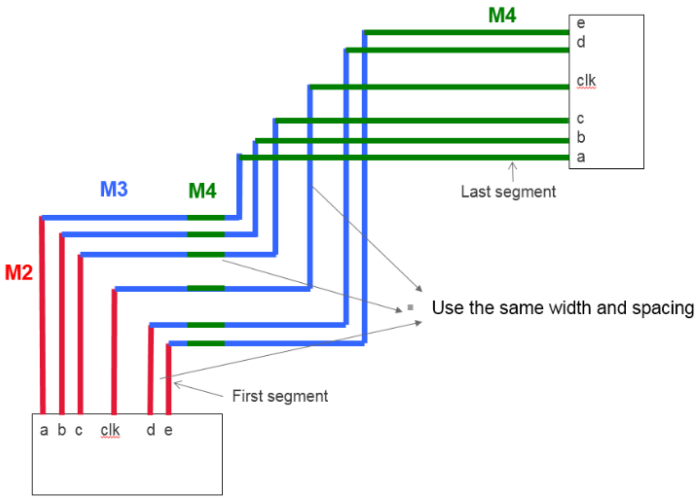
edit_wire_bus_honor_start_parameters

```
edit_wire_bus_honor_start_parameters {true | false}
```

Default: `false`

Data_type: bool, read/write

Specifies that the wire width and spacing of the start points (pins) should be honored when drawing routes for a bus. The following diagram depicts how the width and spacing of the bus is maintained through turns for the the 2nd to N-1 wire segment, if you set `edit_wire_bus_honor_start_parameters` to `true`:



Related Commands

- `delete_routes`
- `edit_add_route_point`
- `edit_merge_routes`
- `edit_move_routes`
- `edit_resize_routes`
- `edit_trim_routes`
- `edit_update_route_layer`
- `edit_update_route_net`
- `edit_update_route_rule`
- `edit_update_route_status`
- `edit_update_via`
- `select_routes`

edit_wire_bus_honor_width_setting

`edit_wire_bus_honor_width_setting {true | false}`

Default: `false` (Pin width is used if `edit_wire_snap_bus_to_pin` is set to `true`)

Data_type: bool, read/write

Specifies whether or not the tool should adjust the wire width when snapping bus wires to pin. By default, if `edit_wire_snap_bus_to_pin` is set to `true`, the tool adjusts the wire width to be the same as the macro or IO pin width. If you do not want the width to be adjusted to match the pin width,

set `edit_wire_bus_honor_width_setting` to `true`. In this case, the tool uses the width value defined by `edit_wire_width_horizontal` or `edit_wire_width_vertical` when snapping bus to pin.

Related Commands

- `delete_routes`
- `edit_add_route_point`
- `edit_merge_routes`
- `edit_move_routes`
- `edit_resize_routes`
- `edit_trim_routes`
- `edit_update_route_layer`
- `edit_update_route_net`
- `edit_update_route_rule`
- `edit_update_route_status`
- `edit_update_via`
- `select_routes`

edit_wire_bus_total_width_horizontal

`edit_wire_bus_total_width_horizontal` *`value`*

Default: `0.0`

Data_type: double, read/write

Specifies the total width of a horizontal bus.

While drawing a bus, the tool auto-sets spacing as:
`wire spacing = (total bus width - total wire width)/ (wire number - 1)`

Related Commands

- delete_routes
- edit_add_route_point
- edit_merge_routes
- edit_move_routes
- edit_resize_routes
- edit_trim_routes
- edit_update_route_layer
- edit_update_route_net
- edit_update_route_rule
- edit_update_route_status
- edit_update_via
- select_routes

edit_wire_bus_total_width_vertical

```
edit_wire_bus_total_width_vertical value
```

Default: 0.0

Data_type: double, read/write

Specifies the total width of a vertical bus.

While drawing a bus, the tool auto-sets spacing as:
```
wire spacing = (total bus width - total wire width)/ (wire number - 1)
```

Related Commands

- delete_routes
- edit_add_route_point
- edit_merge_routes
- edit_move_routes
- edit_resize_routes
- edit_trim_routes
- edit_update_route_layer
- edit_update_route_net
- edit_update_route_rule
- edit_update_route_status
- edit_update_via
- select_routes

edit_wire_change_order_at_turn

```
edit_wire_change_order_at_turn {reverse_order|keep_order}
```

Default: `keep_order`

Data_type: string, read/write

Specifies whether the order of the wires changes (`reverse_order`) or stays the same (`keep_order`) when the wire makes a 90-degree turn.

Related Commands

- delete_routes
- edit_add_route_point
- edit_merge_routes
- edit_move_routes
- edit_resize_routes
- edit_trim_routes
- edit_update_route_layer
- edit_update_route_net
- edit_update_route_rule
- edit_update_route_status
- edit_update_via
- select_routes

edit_wire_check_design_boundary

```
edit_wire_check_design_boundary {true|false}
```

Default: `false`

Data_type: bool, read/write

Specifies that wires and vias are to be made inside the design boundary. If `edit_wire_check_design_boundary` is set to `true`, you cannot manually stretch a wire into the area defined by the `edit_wire_pull_back_distance` attribute.

Note: 45-degree wires do not follow this rule.

Related Commands

- delete_routes
- edit_add_route_point
- edit_merge_routes
- edit_move_routes
- edit_resize_routes
- edit_trim_routes
- edit_update_route_layer
- edit_update_route_net
- edit_update_route_rule
- edit_update_route_status
- edit_update_via
- select_routes

edit_wire_close_polygons

```
edit_wire_close_polygons {true|false}
```

Default: `false`

Data_type: bool, read/write

Specifies whether to close a special route structure toward itself, using the `Escape` key. For the route closure to complete, the ending wire segments must be drawn towards the start wire segments, but do not have to touch them. If you specify this attribute, the software ignores the `edit_wire_extend_wires` attribute.

Related Commands

- delete_routes
- edit_add_route_point
- edit_merge_routes
- edit_move_routes
- edit_resize_routes
- edit_trim_routes
- edit_update_route_layer
- edit_update_route_net
- edit_update_route_rule
- edit_update_route_status
- edit_update_via
- select_routes

edit_wire_color_align_with_track

```
edit_wire_color_align_with_track {true|false}
```

Default: `false`

Data_type: bool, read/write

Specifies whether the wire color is to be changed to match the track color during a move or copy operation. If you set the `edit_wire_color_align_with_track` attribute to `true` and move or copy a wire to snap it to a track, the track color is automatically assigned to the wire. Note that if the wire does not snap to a track or if the wire covers more than one track, this attribute will not have any effect.

This attribute is turned off by default. This means when you move or copy a wire, the wire color is retained by default.

Related Commands

- delete_routes
- edit_add_route_point
- edit_merge_routes
- edit_move_routes
- edit_resize_routes
- edit_trim_routes
- edit_update_route_layer
- edit_update_route_net
- edit_update_route_rule
- edit_update_route_status
- edit_update_via
- select_routes

edit_wire_connect_pin

```
edit_wire_connect_pin {in|out|inout|nodir}
```

Default: `in out inout nodir`

Data_type: enum* {in out inout nodir}, read/write

Specifies the type of pin on which a via should be generated. You can specify the following pin types with this attribute:

- `in` - Generates a via on an input pin.
- `out` - Generates a via on an output pin.
- `inout` - Generates a via on an inout pin.
- `nodir` - Generates a via on a pin for which no direction is specified.

Related Commands

- delete_routes
- edit_add_route_point
- edit_merge_routes
- edit_move_routes
- edit_resize_routes
- edit_trim_routes
- edit_update_route_layer
- edit_update_route_net
- edit_update_route_rule
- edit_update_route_status
- edit_update_via
- select_routes

edit_wire_connect_with_specified_layer

```
edit_wire_connect_with_specified_layer {true|false}
```

Default: `false`

Data_type: bool, read/write

Specifies whether or not the layer of connecting wires should be changed as per the `edit_wire_layer_horizontal` or `edit_wire_layer_vertical` settings when moving wires with the `edit_move_routes` command. If you set this attribute to `true`, `edit_move_routes` honors the `edit_wire_layer_horizontal` or `edit_wire_layer_vertical` settings while jogging metal layers. For instance, suppose you cut a vertical wire on `M4` by using `edit_cut_route` such that the wire is divided into three wire segments. Now, if you select the wire segment in the middle and move it in the horizontal direction, the tool automatically creates new connecting wires on the layer defined by `edit_wire_layer_horizontal` and drops vias correctly. This feature is applicable for both regular and special wires.

Related Commands

- delete_routes
- edit_add_route_point
- edit_merge_routes
- edit_move_routes
- edit_resize_routes
- edit_trim_routes
- edit_update_route_layer
- edit_update_route_net
- edit_update_route_rule
- edit_update_route_status
- edit_update_via
- select_routes

edit_wire_create_crossover_vias

```
edit_wire_create_crossover_vias {true|false}
```

Default: `true`

Data_type: bool, read/write

Specifies whether the software creates a via when you draw a wire that crosses a wire or pin of the same net that is on a different layer. Does not create vias if minimum or maximum layer constraints prevent the creation of vias. If you draw a wire between two pins, the software creates vias on the pins as needed, regardless of whether this option is selected or deselected.

If you set this attribute to `false`, the software does not create vias at crossovers. For example, if you draw a wire from left to right, the software begins creating a via at the left edge of the vertical target. This via is removed as soon as the drawing wire moves past the right edge of the vertical target.

Related Commands

- delete_routes
- edit_add_route_point
- edit_merge_routes
- edit_move_routes
- edit_resize_routes
- edit_trim_routes
- edit_update_route_layer
- edit_update_route_net
- edit_update_route_rule
- edit_update_route_status
- edit_update_via
- select_routes

edit_wire_create_is_edit_flag

```
edit_wire_create_is_edit_flag {true|false}
```

Default: false

Data_type: bool, read/write

Specifies whether to create the db property is_edit for wire edit. This property is used to flag nets with modified wires and vias. This is useful when evaluating how many objects need to be modified for fixing DRC violations and then deciding whether to fix the violations manually or through ECO routing. This property is written to the DEF file.

By default, this attribute is set to false. If you set the value of this attribute to true, the is_edit property is set to 1 for nets with modified wires or vias. The following type of modifications can cause is_edit to be set to 1:

- Drawing a new wire through edit_add_route_point and edit_end_route_point
- Moving a wire/via with edit_move_routes
- Stretching a wire/via with edit_resize_routes

You can reset the is_edit property with set_db.

Note: The is_edit property is independent of edit_wire_keep_status true.

Related Commands

- delete_routes
- edit_add_route_point
- edit_merge_routes
- edit_move_routes
- edit_resize_routes
- edit_trim_routes
- edit_update_route_layer
- edit_update_route_net
- edit_update_route_rule
- edit_update_route_status
- edit_update_via
- select_routes

edit_wire_create_via_on_pin

```
edit_wire_create_via_on_pin {true|false}
```

Default: `true`

Data_type: bool, read/write

Specifies whether the software creates vias at pins. Specify `false` to prevent the software from creating vias at pins. A value of `false` might be needed for bump pins that are not square-shaped.

Related Commands

- delete_routes
- edit_add_route_point
- edit_merge_routes
- edit_move_routes
- edit_resize_routes
- edit_trim_routes
- edit_update_route_layer
- edit_update_route_net
- edit_update_route_rule
- edit_update_route_status
- edit_update_via
- select_routes

edit_wire_cut_class

`edit_wire_cut_class` *`class_name`*

Default: `""`

Data_type: string, read/write

Specifies the cut class name of the vias.

Related Commands

- delete_routes
- edit_add_route_point
- edit_merge_routes
- edit_move_routes
- edit_trim_routes
- edit_update_route_layer
- edit_update_route_net
- edit_update_route_rule
- edit_update_route_status
- edit_update_via
- select_routes

edit_wire_cut_wire_overlap

```
edit_wire_cut_wire_overlap {regular|special}
```

Default: `regular`, which means that regular wires are cut with an overlap while special wires are cut without an overlap

Data_type: enum* {regular special}, read/write

Creates overlaps while cutting specified type of wires. The supported values are:

- `regular` - Cuts regular wires with an overlap
- `special` - Cuts special wires with an overlap

If this attribute is not specified, all wires are cut without overlap. This is equivalent to setting `edit_wire_cut_wire_overlap` to `""`.

Related Commands

- delete_routes
- edit_add_route_point
- edit_merge_routes
- edit_move_routes
- edit_resize_routes
- edit_trim_routes
- edit_update_route_layer
- edit_update_route_net
- edit_update_route_rule
- edit_update_route_status
- edit_update_via
- select_routes

edit_wire_debug_file

`edit_wire_debug_file` *`output_file`*

Default: `""`

Data_type: string, read/write

Specifies the file name for reporting untrimmed wires and vias if `edit_wire_verbose` is set to `true`. If you don't specify a name but `edit_wire_verbose` is set to `true`, the file is named `editTrim_debug.rpt`. If the tool finds the same file in the saved directory, the new file replaces the old one.

Related Commands

- edit_trim_routes
- edit_update_route_status

edit_wire_delete_tsv

`edit_wire_delete_tsv {true|false}`

Default: `false`

Data_type: bool, read/write

Specifies whether or not `delete_routes` deletes the items on a Through-Silicon Via (TSV) layer.

- `true` - The tool deletes the items on TSV layers if `delete_routes` is used without `-layer` or `-selected`.
- `false` - The tool does not delete the items on TSV layers if `delete_routes` is used without `-layer` or `-selected`.

Default: `false`

The `delete_routes` command observes the layer range setting specified by the `edit_wire_layer_min` and `edit_wire_layer_max` attributes, which specify the minimum and maximum routing layers, respectively. If `-layer` is specified with `delete_routes`, the tool ignores the range setting from the `edit_wire_layer_min` and `edit_wire_layer_max` attributes and deletes items only from the specified layers. If `-selected` is specified with `delete_routes`, the tool deletes only the selected wires and vias.

Related Command

- `delete_routes`

edit_wire_delete_pin_with_wire

`edit_wire_delete_pin_with_wire {true|false}`

Default: `true`

Data_type: bool, read/write

Deletes the pin with the wire.

Related Commands

- `delete_routes`
- `edit_add_route_point`
- `edit_merge_routes`
- `edit_move_routes`
- `edit_resize_routes`
- `edit_trim_routes`
- `edit_update_route_layer`
- `edit_update_route_net`
- `edit_update_route_rule`
- `edit_update_route_status`
- `edit_update_via`
- `select_routes`

edit_wire_delete_wire_via_through_layers

```
edit_wire_delete_wire_via_through_layers {true|false}
```

Default: `false`

Data_type: bool, read/write

Deletes wire via in the deep through mode, when set to `true`.

For instance, assume you have two vertical wires `M3` and `M5` at the same location. The tool will not drop vias between these two wires when you draw `M5` because both the wires have the same direction. However, when you draw the horizontal `M6` wire, the tool will drop the vias from `M3` to `M6`. All the three wires have the same net. Now, if you delete the `M6` wire, the setting for `edit_wire_delete_wire_via_through` determines which vias are removed.
If `edit_wire_delete_wire_via_through` is set to `true`, the tool deletes the via from `M3` to `M6`. However, if it is set to `false`, the tool only deletes the via from `M5` to `M6`.

Related Commands

- `delete_routes`
- `edit_add_route_point`
- `edit_merge_routes`
- `edit_move_routes`
- `edit_resize_routes`
- `edit_trim_routes`
- `edit_update_route_layer`
- `edit_update_route_net`
- `edit_update_route_rule`
- `edit_update_route_status`
- `edit_update_via`
- `select_routes`

edit_wire_display_wire_length_with_cursor

```
edit_wire_display_wire_length_with_cursor {true|false}
```

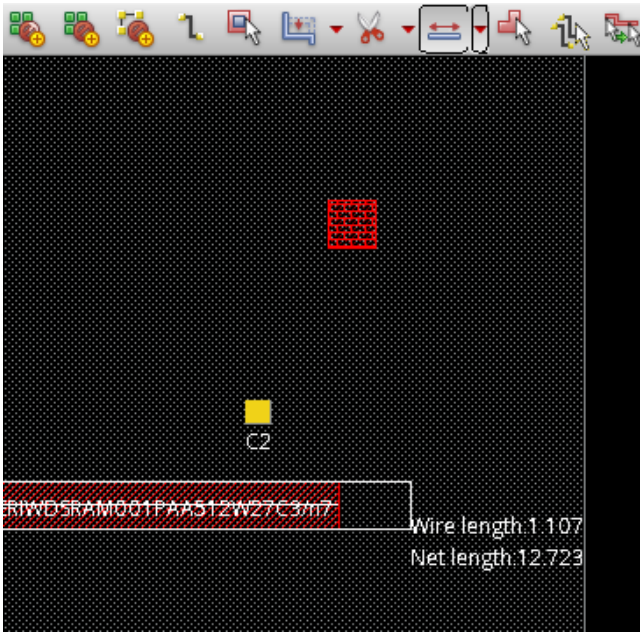
Default: `true`

Data_type: bool, read/write

Specifies whether or not to display the total and current wire length while editing wires by using the mouse. When this parameter is set to `true`, the tool displays the total and current wire length during the following wire edit

operations:

- Adding a new regular, special, or patch wire
- Stretching the length of a regular or special wire
- Resizing a patch wire



edit_wire_draw_shield

```
edit_wire_draw_shield {true|false}
```

Default: `false`

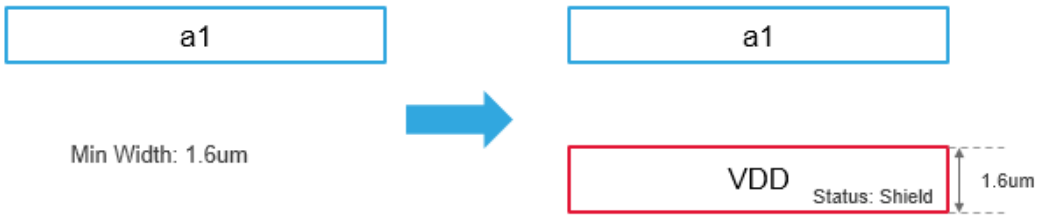
Data_type: bool, read/write

Specifies whether the wire being created is a shield wire.

When `edit_wire_draw_shield` is set to `true`, the wire editor creates a wire with the net name specified using the `shielding_nets` attribute. The newly created wire/via has its status set to `shield` and its shield signal net set to the net name specified with the `edit_wire_nets` attribute. Note that you can specify only one net with the `edit_wire_nets` attribute when drawing a shield wire. The new shield wire/via also has its shape set to `None`. The shield wire defaults to the minimum width and spacing, unless explicitly set to other values. Note that NanoRoute supports only minimum width and minimum spacing.

For example, If you want to draw a shield wire for the wire a1, set the following attributes before drawing the wire:

```
set_db edit_wire_nets a1
set_db edit_wire_shielding_nets VDD
set_db edit_wire_draw_shield true
set_db edit_wire_shape none
set_db edit_wire_width_horizontal 2
```



When `edit_wire_draw_shield` is set to `true`, the tool automatically sets the status of the new wire to `shield` and its width to the minimum width (1.6um). If you also want to change the shield net attribute for the net, set the `edit_wire_update_shield_net` parameter to `true` as well before drawing the wire.

When `edit_wire_draw_shield` is set to `false`, the wire editor reverts to its original behavior and draws a wire/via with the net specified using the `edit_wire_nets` attribute.

Notes:

When `edit_wire_draw_shield` is set to `true`, the following `edit_wire` attributes are ignored:

- `edit_wire_shield`
- `edit_wire_shield_shape`

Related Commands

- `delete_routes`
- `edit_add_route_point`
- `edit_merge_routes`
- `edit_move_routes`
- `edit_resize_routes`
- `edit_trim_routes`
- `edit_update_route_layer`
- `edit_update_route_net`
- `edit_update_route_rule`
- `edit_update_route_status`
- `edit_update_via`
- `select_routes`

edit_wire_draw_with_group_centerline

```
edit_wire_draw_with_group_centerline {true|false}
```

Default: `false`

Data_type: bool, read/write

Draws the bus along the center-line of whole wires when set to `true`.

Enable this option if you set `edit_wire_auto_distribute_bus` to `true`.

If `edit_wire_draw_with_group_centerline` is set to `true`, the tool ignores the `edit_wire_drawing_wire` setting.

Related Commands

- `delete_routes`
- `edit_add_route_point`
- `edit_merge_routes`
- `edit_move_routes`
- `edit_resize_routes`
- `edit_trim_routes`
- `edit_update_route_layer`
- `edit_update_route_net`
- `edit_update_route_rule`
- `edit_update_route_status`
- `edit_update_via`
- `select_routes`

edit_wire_drawing_wire

`edit_wire_drawing_wire` *`value`*

Default: `1`

Data_type: int, read/write

Specifies which of the nets used with the `edit_wire_nets` attribute corresponds to the mouse pointer location when adding an array of wires. For horizontal wires, a value of `1` indicates the bottom-most net. For vertical wires, a value of `1` indicates the left-most net. The *`value`* is an integer of the same or lower value as the number of nets. For example, if you specify 5 nets, you can specify `1`, `2`, `3`, `4`, or `5` as the value for this parameter.

This net is also used as the reference net when changing direction. All other nets around this net are lengthened or shortened accordingly.

Related Commands

- delete_routes
- edit_add_route_point
- edit_merge_routes
- edit_move_routes
- edit_resize_routes
- edit_trim_routes
- edit_update_route_layer
- edit_update_route_net
- edit_update_route_rule
- edit_update_route_status
- edit_update_via
- select_routes

edit_wire_drc_aware_cross_metal

```
edit_wire_drc_aware_cross_metal {true|false}
```

Default: `true, read/write`

Data_type: bool

Specifies whether or not to stretch or move a wire back to the closest location available before a violation occurs with the crossing metal. This option works only if `edit_wire_stop_at_drc` is set to `true`:

- If both `edit_wire_stop_at_drc` and `edit_wire_drc_aware_cross_metal` are set to `true`, the tool stops moving or stretching the wire at the closest location available before a spacing violation occurs with the crossing metal.
- If `edit_wire_stop_at_drc` is set to `true` but `edit_wire_drc_aware_cross_metal` is set to `false`, the tool moves or stretches the wire to the desired location across the crossing metal even if it results in a spacing DRC violation.

This attribute impacts the following type of modifications:

- Drawing a new wire through `edit_add_route_point` and `edit_end_route_point`
- Moving a wire/via with `edit_move_routes`
- Stretching a wire/via with `edit_resize_routes`

Related Commands

- edit_add_route_point
- edit_end_route_point
- edit_move_routes
- edit_resize_routes

edit_wire_drc_on

```
edit_wire_drc_on {true|false}
```

Default: `true`

Data_type: bool, read/write

Specifies whether design rule checking (DRC) should be performed during wire and via editing.

While modifying a via with edit_update_via, the `edit_wire_drc_on` and `edit_wire_keep_via` settings are used in combination to decide whether or not a via should be replaced if the new via cell results in DRC violations. For more details, see edit_update_via.

Related Commands

- delete_routes
- edit_add_route_point
- edit_merge_routes
- edit_move_routes
- edit_resize_routes
- edit_trim_routes
- edit_update_route_layer
- edit_update_route_net
- edit_update_route_rule
- edit_update_route_status
- edit_update_via
- select_routes

edit_wire_drc_use_non_default_spacing

```
edit_wire_drc_use_non_default_spacing {true|false}
```

Default: `false`

Data_type: bool, read/write

Specifies whether non-default spacing defined in the non-default rule (NDR) of the editing net should be used for spacing check during wire editing.

Note

- By default (`false`), the Wire Editor DRC check honors the default/minimum spacing.
- If the net or via has an NDR with the `HARDSPACING` keyword, the Wire Editor DRC check honors the non-default spacing even if `edit_wire_use_non_default_spacing` is set to `false`.
- After `edit_wire_use_non_default_spacing 1` is set, all subsequent wire editing commands honor the non-default spacing during the geometry check on the net/via if the NDR is defined.

Related Commands

- `delete_routes`
- `edit_add_route_point`
- `edit_merge_routes`
- `edit_move_routes`
- `edit_resize_routes`
- `edit_trim_routes`
- `edit_update_route_layer`
- `edit_update_route_net`
- `edit_update_route_rule`
- `edit_update_route_status`
- `edit_update_via`
- `select_routes`

edit_wire_extend_wires

```
edit_wire_extend_wires {start|end|start_cell_boundary|end_cell_boundary}
```

Default: `""`

Data_type: enum* {start end start_cell_boundary end_cell_boundary}, read/write

Specifies which boundary of the wire segment extends and connects to the first logical target, after completing a signal or power route. You can specify the following boundary options with this parameter:

- `start` - Specifies that the startpoint of the route segment extends and connects to the first logical target, after

completing a signal or power route. If there is no logical target, the startpoint is not changed.

- `end` - Specifies that the endpoint of the route segment extends and connects to the first logical target, after completing a signal or power route. If there is no logical target, the endpoint is not changed.
- `start_cell_boundary` - Specifies that the startpoint of the segment always extends to the cell boundary, after completing a special route. If the segment is extended to the boundary and crosses the wires on the same net, the tool makes a via connection to the wires. In addition, the tool creates pins at the boundary. This parameter can only be used in conjunction with `-extend_wires start`.
- `end_cell_boundary` - Specifies that the endpoint of the segment always extends to the cell boundary, after completing a special route. If the segment is extended to the boundary and crosses wires on the same net, the tool makes a via connection to the wires. In addition, the tool creates pins at the boundary. This parameter can only be used in conjunction with `-extend_wires end`.

Note: If you do not want the wire segment to be extended, set `edit_wire_extend_wires` to `""`.

Related Commands

- `delete_routes`
- `edit_add_route_point`
- `edit_merge_routes`
- `edit_move_routes`
- `edit_resize_routes`
- `edit_trim_routes`
- `edit_update_route_layer`
- `edit_update_route_net`
- `edit_update_route_rule`
- `edit_update_route_status`
- `edit_update_via`
- `select_routes`

edit_wire_final_check_with_verify

```
edit_wire_final_check_with_verify {true|false}
```

Default: `false`

Data_type: bool, read/write

Specifies whether to do final checks with verify.

When `edit_wire_final_check_with_verify` is set to `true`, Wire Editor calls the Full Geometry Checker (FGC) engine for the final DRC check on the bounding box of changed area after commits. This ensures that the final wire editing result is DRC clean as per FGC. Specify this attribute if you are using advanced rules, such as 20nm and 14nm.

Related Commands

- `delete_routes`
- `edit_add_route_point`
- `edit_merge_routes`
- `edit_move_routes`
- `edit_resize_routes`
- `edit_trim_routes`
- `edit_update_route_layer`
- `edit_update_route_net`
- `edit_update_route_rule`
- `edit_update_route_status`
- `edit_update_via`
- `select_routes`

edit_wire_ignore_drc

`edit_wire_ignore_drc {conn_antenna|max_via_stack|min_cut|min_enclosed_area|min_step|protrusion}`

Default: `""`

Data_type: enum* {conn_antenna max_via_stack min_cut min_enclosed_area min_step protrusion}, read/write

Specifies which violation type should be ignored during wire editing. You can specify the following options:

- `conn_antenna` - Does not check the connection antenna for DRC violations.
- `max_via_stack` - Does not check maximum via stack DRC violations during wire editing.
- `min_cut` - Does not check for minimum cut DRC during wire editing.
- `min_enclosed_area` - Does not perform minimum enclosed area DRC during wire editing.
- `min_step` - Does not perform minimum step DRC during wire editing.
- `protrusion` - Does not perform protrusion DRC during wire editing.

Note: If you do not want to ignore any violation type, set `edit_wire_ignore_drc` to `""`.

Related Commands

- delete_routes
- edit_add_route_point
- edit_merge_routes
- edit_move_routes
- edit_resize_routes
- edit_trim_routes
- edit_update_route_layer
- edit_update_route_net
- edit_update_route_rule
- edit_update_route_status
- edit_update_via
- select_routes

edit_wire_jog_connect_layer

`edit_wire_jog_connect_layer` *value*

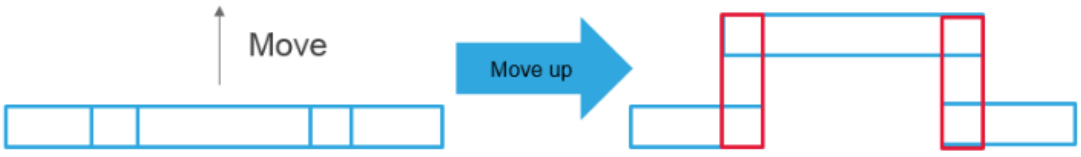
Default: `0`

Data_type: int, read/write

Specifies the number of layers by which the new jog wire segments created on moving a wire should be moved up or down. Note that the `edit_wire_jog_connect_layer` attribute will have no impact if you have set the `edit_wire_connect_with_specified_layer` attribute to `true`.

- The following example uses the `edit_wirejog_connect_layer` attribute to specify that the tool should automatically move up new jog wires created on moving a wire segment by 1 layer:

  ```
  set_db edit_wire_connect_with_specified_layer false
  set_db edit_wire_jog_connect_layer 1
  ```

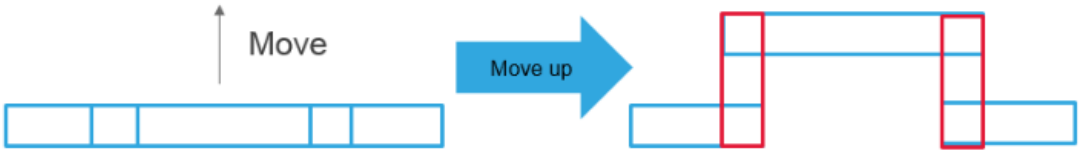
  

  Here, the blue wire segments are on layer `M2` while the red jog wire segments are on the `M3` layer.

- In the following example, as the `edit_wire_jog_connect_layer` parameter is set to -1, the tool automatically moves down the new jog wires by 1 layer:

  ```
  set_db edit_wire_connect_with_specified_layer false
  ```

```
set_db edit_wire_jog_connect_layer -1
```



In this case, the blue wire segments are on layer M2 while the red jog wire segments are on the M1 layer.

Related Commands

- `delete_routes`
- `edit_add_route_point`
- `edit_merge_routes`
- `edit_move_routes`
- `edit_resize_routes`
- `edit_trim_routes`
- `edit_update_route_layer`
- `edit_update_route_net`
- `edit_update_route_rule`
- `edit_update_route_status`
- `edit_update_via`
- `select_routes`

edit_wire_keep_floating_via

```
edit_wire_keep_floating_via {true|false}
```

Default: `false`

Data_type: bool, read/write

Controls whether or not floating vias are retained when wires are deleted. By default, floating vias are deleted when you delete a wire.

Set the `edit_wire_keep_floating_via` attribute to `true` if you want to retain floating vias when you delete wires.

Related Commands

- delete_routes
- edit_add_route_point
- edit_merge_routes
- edit_move_routes
- edit_resize_routes
- edit_trim_routes
- edit_update_route_layer
- edit_update_route_net
- edit_update_route_rule
- edit_update_route_status
- edit_update_via
- select_routes

edit_wire_keep_status

```
edit_wire_keep_status {true|false}
```

Default: `true`

Data_type: bool, read/write

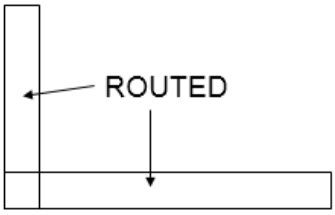
Specifies whether or not the object status should be changed during editing. This option is set to `true` by default, which means wire status is not changed after editing. If this option is set to `false`, the wire status is changed to `FIXED` automatically after `edit_move_routes` and `edit_resize_routes`.

Notes:

This option has an impact on both regular and special wires and vias. The tool only changes the status of the wire segment that is modified.

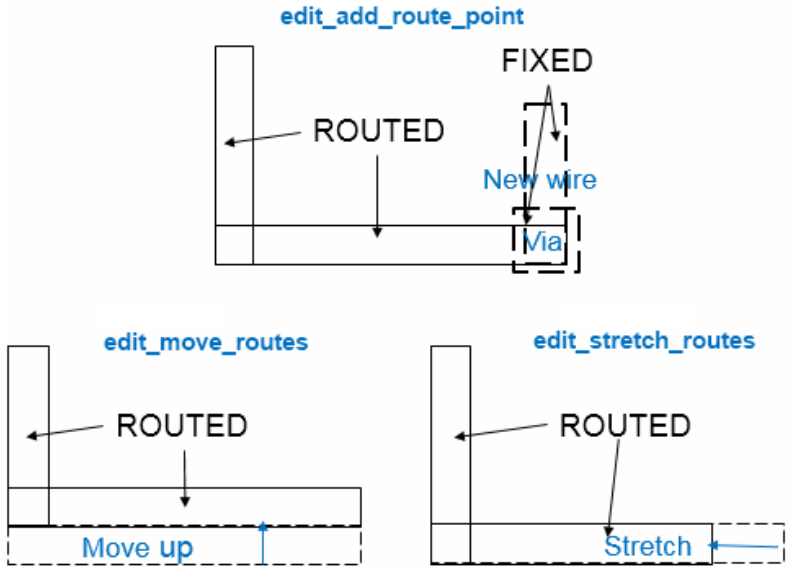
Use the `set_db edit_wire_status FIXED` command to set the status of newly drawn wires to `FIXED` after `edit_add_route_point`.

Consider the wire segment shown in the figure below:



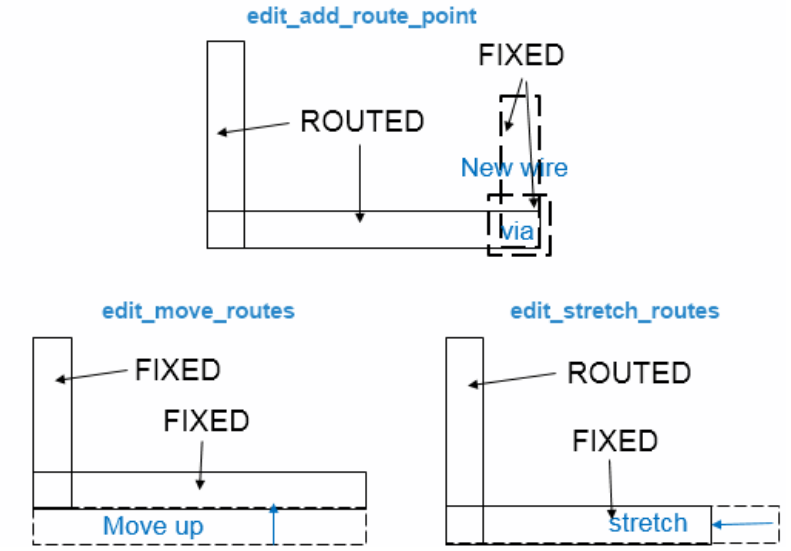
- The following command specifies that the status of the modified wire segment should be retained after `edit_move_routes` and `edit_resize_routes`:

  ```
  set_db edit_wire_status FIXED
  set_db edit_wire_keep_status true
  ```



- The following command specifies that the status of the modified wire segment should be changed to `FIXED` after `edit_move_routes` and `edit_resize_routes`:

  ```
  set_db edit_wire_status FIXED
  set_db edit_wire_keep_status false
  ```



Related Commands

- delete_routes
- edit_add_route_point
- edit_merge_routes
- edit_move_routes
- edit_resize_routes
- edit_trim_routes
- edit_update_route_layer
- edit_update_route_net
- edit_update_route_rule
- edit_update_route_status
- edit_update_via
- select_routes

edit_wire_keep_via

```
edit_wire_keep_via {true|false}
```

Default: `false`

Data_type: bool, read/write

Controls whether or not the original via is retained in a wire move or stretch. If this attribute is set to `true`, the tool keeps the via unchanged if there is no DRC violation as a result of the move or stretch. However, the tool can change the via if a DRC violation occurs. Set the `edit_wire_keep_via` attribute to `false` if you do not want the original via to be retained when a wire is moved or stretched. This attribute is applicable for both regular and special vias.

While modifying a via with `edit_update_via`, the `edit_wire_keep_via` and `edit_wire_drc_on` settings are used in combination to decide whether or not a via should be replaced if the new via cell results in DRC violations. For more details, see `edit_update_via`.

Related Commands

- delete_routes
- edit_add_route_point
- edit_merge_routes
- edit_move_routes
- edit_resize_routes
- edit_trim_routes
- edit_update_route_layer
- edit_update_route_net
- edit_update_route_rule
- edit_update_route_status
- edit_update_via
- select_routes

edit_wire_lateral_movement_range

`edit_wire_lateral_movement_range` *value*

Default: `0.0`

Data_type: double, read/write

Specifies how much the wire you are adding or stretching can be moved laterally to avoid potential DRC violations with existing wires, nets, or blockages. The preferred lateral movement direction is up for horizontal wires and left for vertical wires, provided this movement does not result in a new violation.

If a violation cannot be avoided even by moving the new wire within the specified lateral movement range, the tool obeys other specified wire edit settings. For example, if the `edit_wire_stop_at_drc 1` setting is specified, the tool moves or stretches the new wire up till the closest location available before the DRC violation occurs.

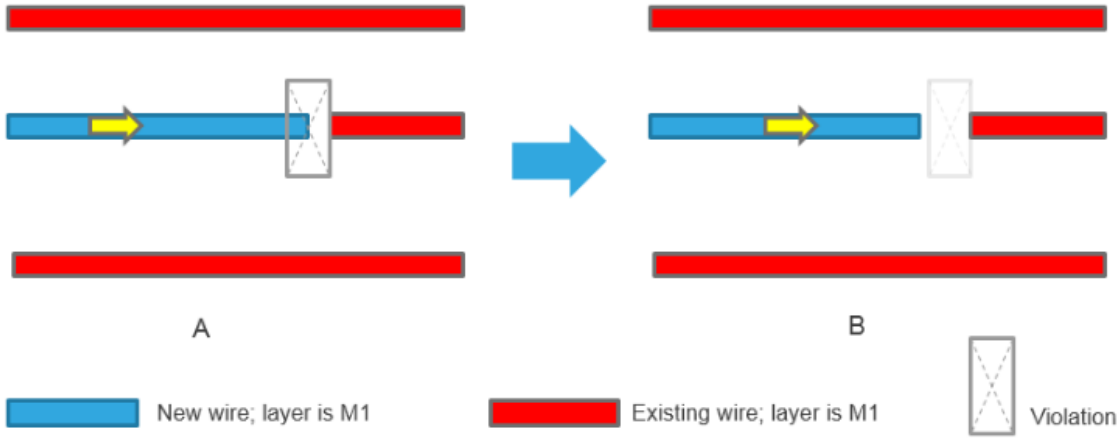
Notes:

- The `edit_wire_lateral_movement_range` attribute works only if `edit_wire_drc_on` is set to `1`.
  If `edit_wire_drc_on` is set to `0`, the tool ignores the `edit_wire_lateral_movement_range` setting.
- The `edit_wire_lateral_movement_range` attribute can support multiple nets.
- The `edit_wire_lateral_movement_range` attribute is relevant only for avoiding violations with wires and vias. If the wire being added crosses a routing blockage or cell OBS, the `edit_wire_lateral_movement_range` setting is ignored.

See the examples below to understand how the `edit_wire_lateral_movement_range` attribute works:

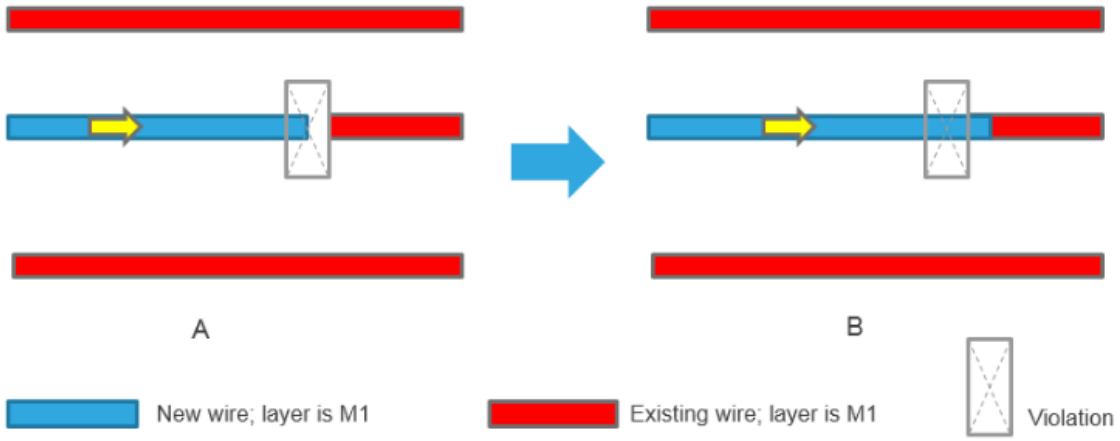
In the following examples, the blue wire is being moved or stretched from left to right. See how the tool behavior changes with the various edit_wire attribute settings given below:

- ```
  set_db edit_wire_drc_on 1
  set_db edit_wire_lateral_movement_range 0
  set_db edit_wire_stop_at_drc 1
  ```



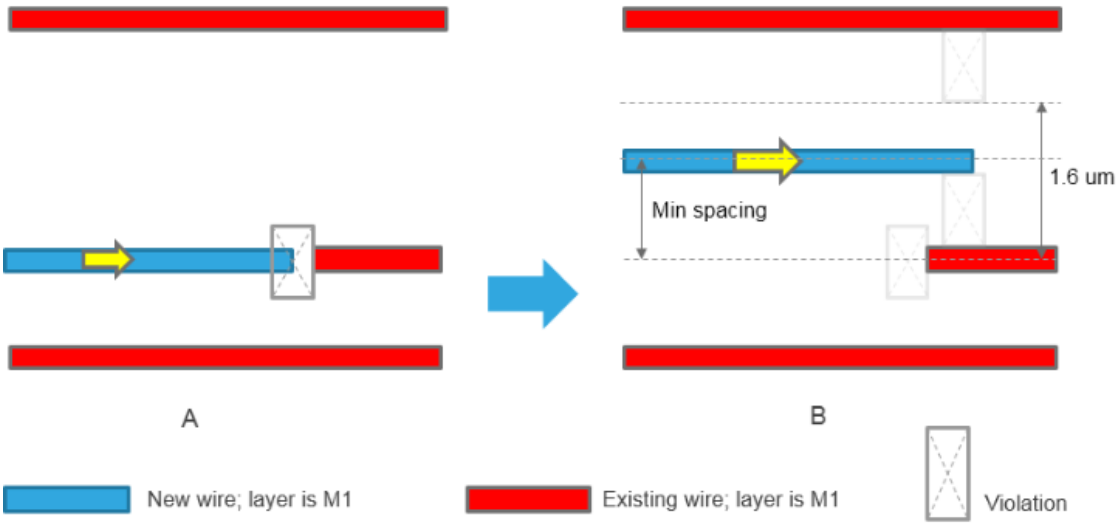
  This wire edit will generate a violation with an existing wire (see A), so the tool shrinks the new wire to a distance such that there is no violation (see B).

- ```
  set_db edit_wire_drc_on 1
  set_db edit_wire_lateral_movement_range 0
  set_db edit_wire_stop_at_drc 0
  ```



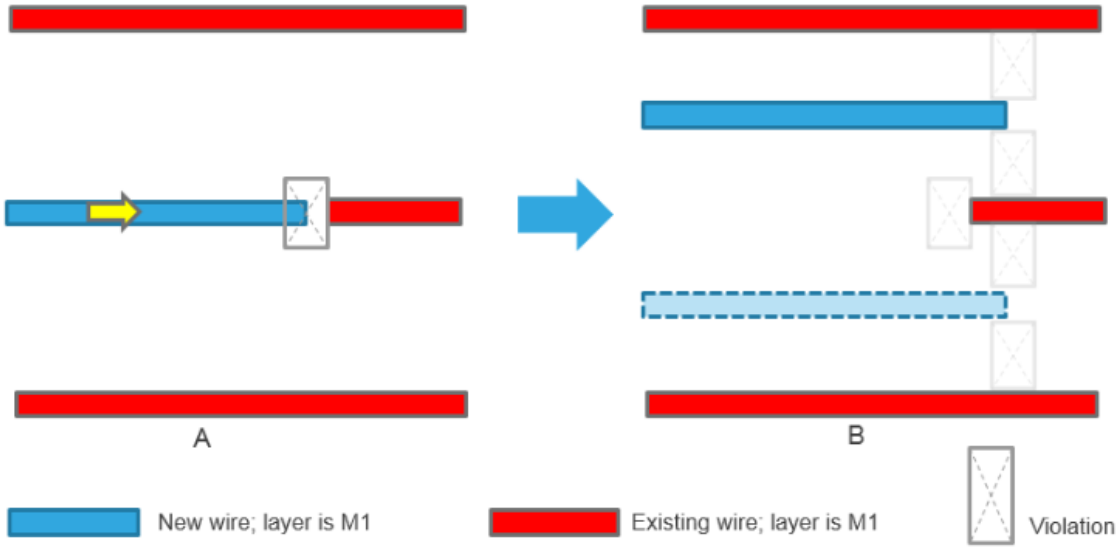
  This wire edit will generate a violation with an existing wire (see A). However, the tool allows the new wire to be moved or stretched to any length and reports a violation because of the `edit_wire_stop_at_drc 0` setting (see B).

- ```
  set_db edit_wire_drc_on 1
  set_db edit_wire_lateral_movement_range 1.6
  set_db edit_wire_stop_at_drc 1
  ```
  ```



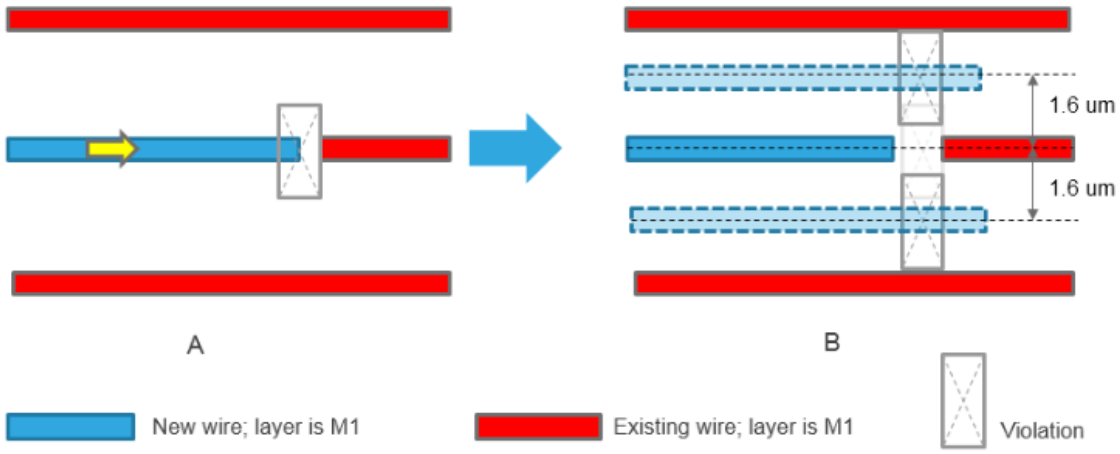
This wire edit will generate a violation with an existing wire (see A). So, the tool moves the new wire up by a minimum distance to avoid generating a violation (see B).

- ```
  set_db edit_wire_drc_on 1
  set_db edit_wire_lateral_movement_range 1.6
  set_db edit_wire_stop_at_drc 1
  ```



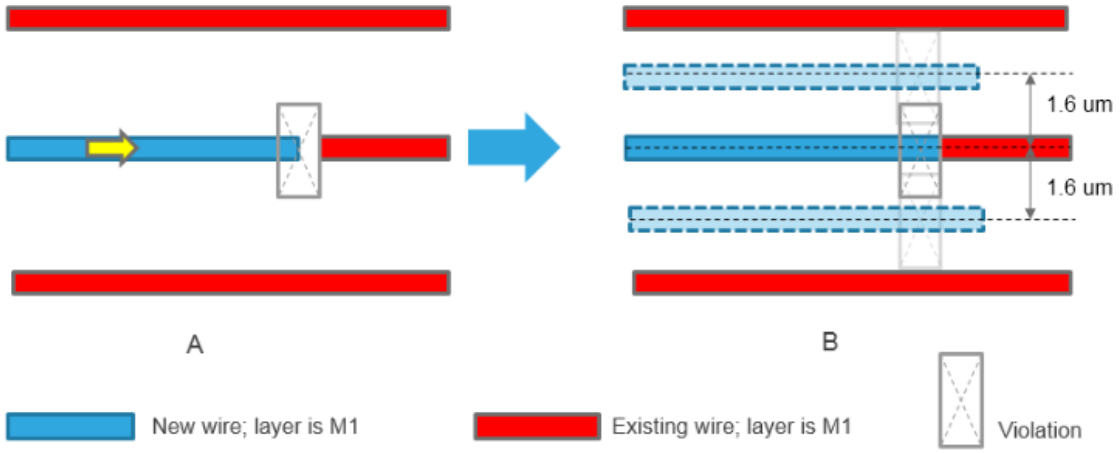
This wire edit will generate a violation with an existing wire (see A). The blue wire can be moved either up or down to avoid the violation. In this case, the tool moves the wire up, which is the preferred direction for horizontal wires (see B).

- ```
  set_db edit_wire_drc_on 1
  set_db edit_wire_lateral_movement_range 1.6
  set_db edit_wire_stop_at_drc 1
  ```



This wire edit will generate a violation with an existing wire (see A). However, even if the blue wire is moved up or down within the specified lateral movement range, the tool still cannot avoid a violation. Therefore, the tool obeys other specified wire edit settings. In this example, the tool obeys the `-stop_at_drc 1` setting and moves or stretches the blue wire up till the closest location available before the DRC violation occurs (see B).

- ```
  set_db edit_wire_drc_on 1
  set_db edit_wire_lateral_movement_range 1.6
  set_db edit_wire_stop_at_drc 0
  ```



This wire edit will generate a violation with an existing wire (see A). However, even if the blue wire is moved up or down within the specified lateral movement range, the tool still cannot avoid a violation. Therefore, the tool obeys other specified wire edit settings. In this example, the tool obeys the `-stop_at_drc 0` setting and allows you to move or stretch the wire even though it results in a DRC violation. (see B).
  ```

Related Commands

- delete_routes
- edit_add_route_point
- edit_merge_routes
- edit_move_routes
- edit_resize_routes
- edit_trim_routes
- edit_update_route_layer
- edit_update_route_net
- edit_update_route_rule
- edit_update_route_status
- edit_update_via
- select_routes

edit_wire_layer

`edit_wire_layer` *name*

Default: `M1` (the first routing layer)

Data_type: string, read/write

Specifies the layer for patch or polygon wires.

Related Commands

- delete_routes
- edit_add_route_point
- edit_merge_routes
- edit_move_routes
- edit_resize_routes
- edit_trim_routes
- edit_update_route_layer
- edit_update_route_net
- edit_update_route_rule
- edit_update_route_status
- edit_update_via
- select_routes

edit_wire_layer_horizontal

`edit_wire_layer_horizontal` *name*

Default: `M1` (the first routing layer)

Data_type: string, read/write

Specifies the layer for horizontal wires. This option supports backside layers as well.

Related Commands

- delete_routes
- edit_add_route_point
- edit_merge_routes
- edit_move_routes
- edit_resize_routes
- edit_trim_routes
- edit_update_route_layer
- edit_update_route_net
- edit_update_route_rule
- edit_update_route_status
- edit_update_via
- select_routes

edit_wire_layer_max

`edit_wire_layer_max` *name*

Default: UNINITIALIZED

Data_type: string, read/write

Specifies the maximum layer for routing. The tool does not create wires above the specified layer. This option supports backside layers as well.

Related Commands

- delete_routes
- edit_add_route_point
- edit_merge_routes
- edit_move_routes
- edit_resize_routes
- edit_trim_routes
- edit_update_route_layer
- edit_update_route_net
- edit_update_route_rule
- edit_update_route_status
- edit_update_via
- select_routes

edit_wire_layer_min

`edit_wire_layer_min` *name*

Default: UNINITIALIZED

Data_type: string, read/write

Specifies the minimum layer for routing. The tool does not create wires below the specified layer. This option supports backside layers as well.

Related Commands

- delete_routes
- edit_add_route_point
- edit_merge_routes
- edit_move_routes
- edit_resize_routes
- edit_trim_routes
- edit_update_route_layer
- edit_update_route_net
- edit_update_route_rule
- edit_update_route_status
- edit_update_via
- select_routes

edit_wire_layer_vertical

`edit_wire_layer_vertical` *name*

Default: `M2` (the second routing layer)

Data_type: string, read/write

Specifies the layer for vertical wires. This option supports backside layers as well.

Related Commands

- delete_routes
- edit_add_route_point
- edit_merge_routes
- edit_move_routes
- edit_resize_routes
- edit_trim_routes
- edit_update_route_layer
- edit_update_route_net
- edit_update_route_rule
- edit_update_route_status
- edit_update_via
- select_routes

edit_wire_look_down_layers

`edit_wire_look_down_layers` *integer*

Default: `100` (all layers)

Data_type: int, read/write

Specifies the number of layers below the current layer that added wires will connect to with a via.

Related Commands

- delete_routes
- edit_add_route_point
- edit_merge_routes
- edit_move_routes
- edit_resize_routes
- edit_trim_routes
- edit_update_route_layer
- edit_update_route_net
- edit_update_route_rule
- edit_update_route_status
- edit_update_via
- select_routes

edit_wire_look_up_layers

`edit_wire_look_up_layers` *integer*

Default: `100` (all layers)

Data_type: int, read/write

Specifies the number of layers above the current layer that added wires will connect to with a via.

Related Commands

- delete_routes
- edit_add_route_point
- edit_merge_routes
- edit_move_routes
- edit_resize_routes
- edit_trim_routes
- edit_update_route_layer
- edit_update_route_net
- edit_update_route_rule
- edit_update_route_status
- edit_update_via
- select_routes

edit_wire_max_pointer_number

edit_wire_max_pointer_number *integer*

Default: 100

Data_type: int, read/write

Specifies the maximum number of pointers to be returned by wire edit commands. Use this attribute along with edit_wire_return_obj_pointer to control object pointers returned by wire edit commands. The returned pointers make it possible for you to select and further edit the newly created or modified wire or via.

Related Commands

- delete_routes
- edit_add_route_point
- edit_merge_routes
- edit_move_routes
- edit_resize_routes
- edit_trim_routes
- edit_update_route_layer
- edit_update_route_net
- edit_update_route_rule
- edit_update_route_status
- edit_update_via
- select_routes

edit_wire_mirror_bus_route

```
edit_wire_mirror_bus_route {true|false}
```

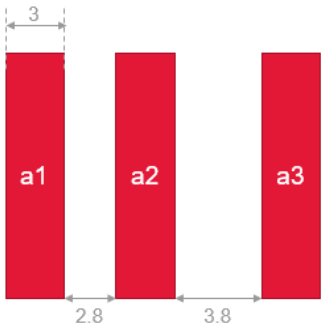
Default: `false`

Data_type: bool, read/write

Specifies whether or not to reverse the net order for the bus route. When this attribute is specified, the definition of the wire override spec is reversed. For example, suppose you have drawn a bus route with the following settings:

```
set_db edit_wire_nets {a1 a2 a3}
set_db edit_wire_override true
set_db edit_wire_wire_override_spec {{1 3 2.8} {2 3 3.8} {3 3 4.8}}
```

The bus is drawn as follows, with nets counted from left to right:



Now, by setting `edit_wire_mirror_bus_route` to `true`, you can draw a mirror bus route with nets counted from right to

left:

```
set_db edit_wire_nets {a1 a2 a3}
set_db edit_wire_override true
set_db edit_wire_wire_override_spec {{1 3 2.8} {2 3 3.8} {3 3 4.8}}
set_db edit_wire_mirror_bus_route true
```

Related Commands

- delete_routes
- edit_add_route_point
- edit_merge_routes
- edit_move_routes
- edit_resize_routes
- edit_trim_routes
- edit_update_route_layer
- edit_update_route_net
- edit_update_route_rule
- edit_update_route_status
- edit_update_via
- select_routes

edit_wire_nets

`edit_wire_nets` *nets*

Default: ""

Data_type: string, read/write

Specifies one or more nets for editing. If you specify multiple nets, separate each net name with a space and enclose the names in curly braces.

You can use the following format to specify a bus:

netName`[`*x*`:`*y*`]`

For example, to draw eight wires for a bus, type the following:

`edit_wire_nets` *netName*`[7:0]`

This is equivalent to specifying *netName*`[7], `*netName*`[6], ... `*netName*`[0]`. Specifying `edit_wire_nets` *netName*`[0:7]` reverses the order.

The order in which you specify the nets determines the drawing arrangement of the wires.

- For horizontally drawn wires, the first net specified is drawn as the bottom-most wire, followed by the next net on its top side, and so on. The last net specified is drawn as the top-most wire.
- For vertically drawn wires, the first net specified is drawn as the left-most wire, followed by the next net, and so on. The last net specified is drawn as the right-most wire.

Related Commands

- `delete_routes`
- `edit_add_route_point`
- `edit_merge_routes`
- `edit_move_routes`
- `edit_resize_routes`
- `edit_trim_routes`
- `edit_update_route_layer`
- `edit_update_route_net`
- `edit_update_route_rule`
- `edit_update_route_status`
- `edit_update_via`
- `select_routes`

edit_wire_no_merge_special_wire

`edit_wire_no_merge_special_wire {true|false}`

Default: `false`

Data_type: bool, read/write

Specifies whether or not a wire being added should be merged automatically with any existing wire at that point. Set this attribute to true, if you want to disable the auto merge feature.

Related Commands

- delete_routes
- edit_add_route_point
- edit_merge_routes
- edit_move_routes
- edit_resize_routes
- edit_trim_routes
- edit_update_route_layer
- edit_update_route_net
- edit_update_route_rule
- edit_update_route_status
- edit_update_via
- select_routes

edit_wire_only_show_edit_layer

```
edit_wire_only_show_edit_layer {true|false}
```

Default: `true`

Data_type: bool, read/write

Specifies that only the layer with the objects being edited should be shown and other layers and objects should be dimmed out during the following wire/via edit operations:

- Add
- Resize
- Copy/Move

Related Commands

- delete_routes
- edit_add_route_point
- edit_merge_routes
- edit_move_routes
- edit_resize_routes
- edit_trim_routes
- edit_update_route_layer
- edit_update_route_net
- edit_update_route_rule
- edit_update_route_status
- edit_update_via
- select_routes

edit_wire_orthogonal_connection_only

```
edit_wire_orthogonal_connection_only {true|false}
```

Default: `true`

Data_type: bool, read/write

Specifies whether to consider wires and pins that are in the same direction as a target for connection. For example, if you extend a wire to the left, any horizontal wire it meets is not considered for a target if the value of this attribute is set to `true`. However, a vertical wire is considered a target.

Related Commands

- delete_routes
- edit_add_route_point
- edit_merge_routes
- edit_move_routes
- edit_resize_routes
- edit_trim_routes
- edit_update_route_layer
- edit_update_route_net
- edit_update_route_rule
- edit_update_route_status
- edit_update_via
- select_routes

edit_wire_outer_shield_spacing

`edit_wire_outer_shield_spacing` *value*

Default: `0.0` (minimum spacing on the layer)

Data_type: double, read/write

Specifies a spacing value in microns to be used by the `edit_wire_shield low` and `edit_wire_shield high` attribute settings.

Related Commands

- delete_routes
- edit_add_route_point
- edit_merge_routes
- edit_move_routes
- edit_resize_routes
- edit_trim_routes
- edit_update_route_layer
- edit_update_route_net
- edit_update_route_rule
- edit_update_route_status
- edit_update_via
- select_routes

edit_wire_outer_shield_width

`edit_wire_outer_shield_width` *value*

Default: `0.0` (minimum wire width on the layer)

Data_type: double, read/write

Specifies a width value in microns to be used for outer shield(s) depending on the setting of the `edit_wire_shield` attribute.

Note: You must define the outer shield first. The outer shield is the shielding wire beyond the first or the last wire being created manually.

Related Commands

- delete_routes
- edit_add_route_point
- edit_merge_routes
- edit_move_routes
- edit_resize_routes
- edit_trim_routes
- edit_update_route_layer
- edit_update_route_net
- edit_update_route_rule
- edit_update_route_status
- edit_update_via
- select_routes

edit_wire_override

```
edit_wire_override {true|false}
```

Default: `false`

Data_type: bool, read/write

Specifies whether to use the override specification.

Related Commands

- delete_routes
- edit_add_route_point
- edit_merge_routes
- edit_move_routes
- edit_resize_routes
- edit_trim_routes
- edit_update_route_layer
- edit_update_route_net
- edit_update_route_rule
- edit_update_route_status
- edit_update_via
- select_routes

edit_wire_partial_overlap_threshold

`edit_wire_partial_overlap_threshold` *value*

Default: `0`

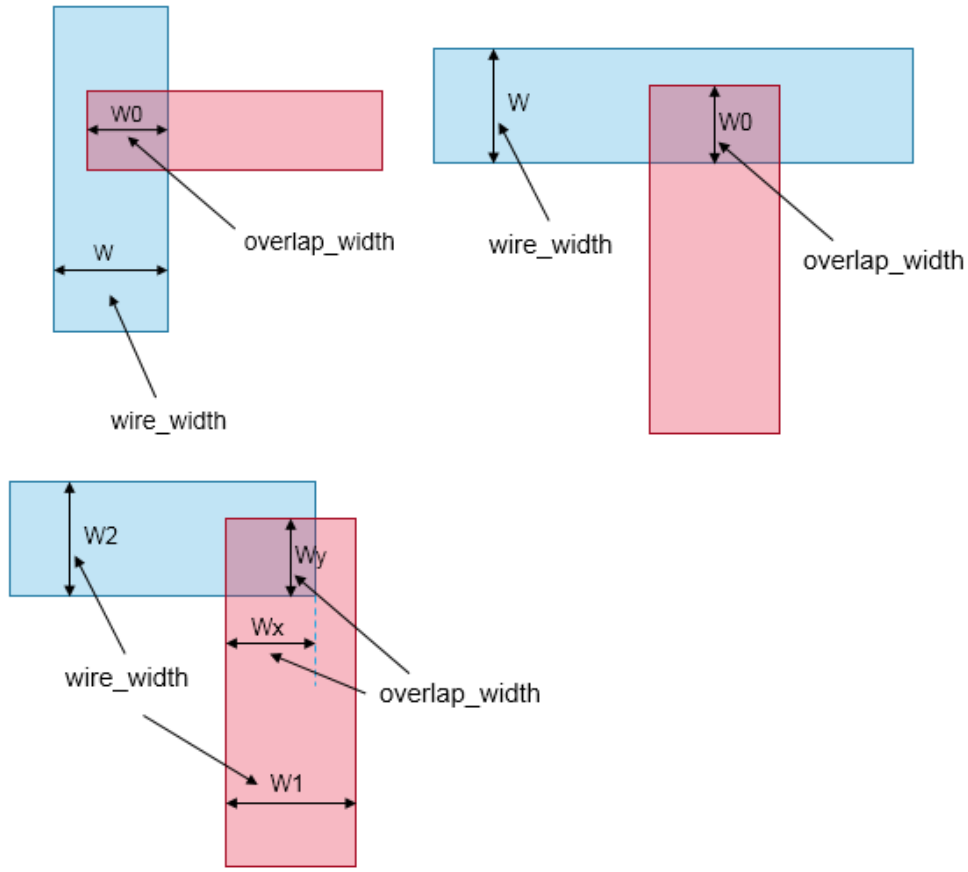
Data_type: int, read/write

Specifies the ratio threshold for special via creation on partially overlapping wires. The threshold value may range from `0` to `100`, with a value of `100` specifying that no via is dropped on a partial overlap intersection. The tool drops a via only when the `overlap_width` to `wire_width` ratio percentage is greater than or equal to the specified threshold value.

For example, suppose the threshold value is set to `100`. Now consider the following cases in which the pink wire is being drawn over the existing blue wire:

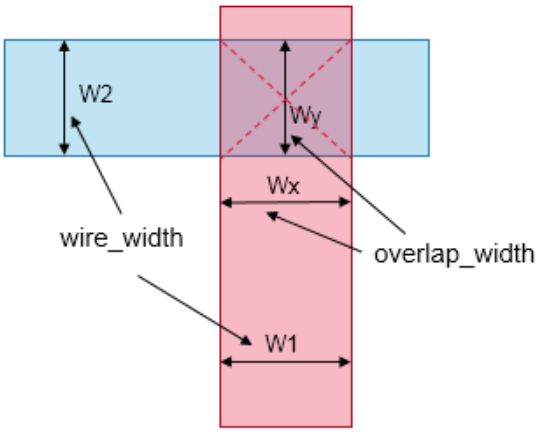
Case 1 - Overlap width less than wire width

In each of the cases depicted below, the `overlap_width` is less than the corresponding `wire_width`. As the `overlap_width` to `wire_width` ratio percentage is less than `100`, which is the specified threshold value, the tool does not drop a via on the partial overlap area:



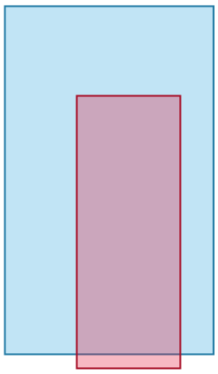
Case 2 - Overlap width equal to wire width

In the following case, the `overlap_width` is equal to the corresponding `wire_width`. As the `overlap_width` to `wire_width` ratio percentage is equal to `100`, which is the specified threshold value, the tool drops a via on the overlap area.



Notes:

- The `partial_overlap_threshold` attribute value is applicable only if the new wire is crossing an existing wire in a different direction. If the new wire being drawn is overlapping an existing wire in the same direction, the tool does not create a via:



- If the `edit_wire_create_crossover_vias` attribute is set to `false`, the tool ignores the setting of the `edit_wire_partial_overlap_threshold` attribute and does not create any via on the overlap area.

Related Commands

- `delete_routes`
- `edit_add_route_point`
- `edit_merge_routes`
- `edit_move_routes`
- `edit_resize_routes`
- `edit_trim_routes`
- `edit_update_route_layer`
- `edit_update_route_net`
- `edit_update_route_rule`
- `edit_update_route_status`
- `edit_update_via`
- `select_routes`

edit_wire_pull_back_distance

`edit_wire_pull_back_distance` *`string`*

Default: `auto`, which means the tool uses the largest min spacing of the layer as the forbidden distance.

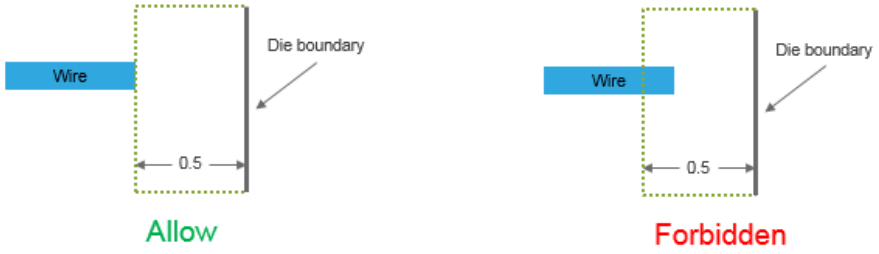
Data_type: string, read/write

Specifies the value for forbidden distance. If the `edit_wire_check_design_boundary` attribute is set to `true`, the tool takes into account the settings of the `edit_wire_pull_back_distance` attribute during wire edit operations.

If the `edit_wire_pull_back_distance` value is positive, the tool marks the corresponding area inside the boundary as an extra keep-out area. For example, if you specify the following command, the tool marks 0.5 distance inside the die boundary on all edges as an extra keep-out area and does not allow you to stretch a wire into this area. You are allowed to stretch the wire only up till the forbidden line.

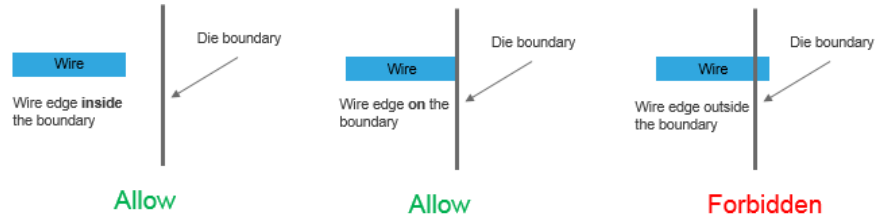
```
set_db edit_wire_check_design_boundary true

set_db edit_wire_pull_back_distance 0.5
```



If the `edit_wire_pull_back_distance` value is `0`, the tool does not allow you to manually stretch a wire beyond the die boundary. If you try to stretch the wire edge beyond the boundary, the tool stops the wire edge at the boundary and prints a warning.

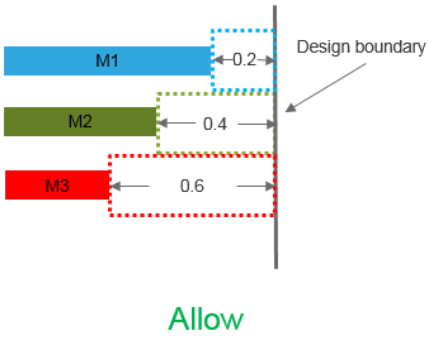
```
set_db edit_wire_check_design_boundary true
set_db edit_wire_pull_back_distance 0
```



If the edge of a wire is already beyond the boundary (with `set_db edit_wire_check_design_boundary true` and `set_db edit_wire_pull_back_distance 0`) and you attempt to stretch it further, the wire remains in the current location and the tool does not allow further stretching.

If you stretch multiple wires with `edit_wire_pull_back_distance` set to `auto`, the tool uses the largest spacing value of each layer as the pull back distance for related wire. For example, the diagram below illustrates pull back distances if the spacing for different layers is set as follows:

```
M1: 0.2      M2: 0.4      M3: 0.6
```



If you do not need the `edit_wire_pull_back_distance` constraint, set `edit_wire_check_design_boundary` to `0`.

Note: 45-degree wires do not follow this rule

Related Commands

- `delete_routes`
- `edit_add_route_point`
- `edit_merge_routes`
- `edit_move_routes`
- `edit_resize_routes`
- `edit_trim_routes`
- `edit_update_route_layer`
- `edit_update_route_net`
- `edit_update_route_rule`
- `edit_update_route_status`
- `edit_update_via`
- `select_routes`

edit_wire_reshape

`edit_wire_reshape {true|false}`

Default: `false`

Data_type: bool, read/write

Specifies that when you add new wires, existing redundant wires within the route are automatically removed.

Related Commands

- delete_routes
- edit_add_route_point
- edit_merge_routes
- edit_move_routes
- edit_resize_routes
- edit_trim_routes
- edit_update_route_layer
- edit_update_route_net
- edit_update_route_rule
- edit_update_route_status
- edit_update_via
- select_routes

edit_wire_return_obj_pointer

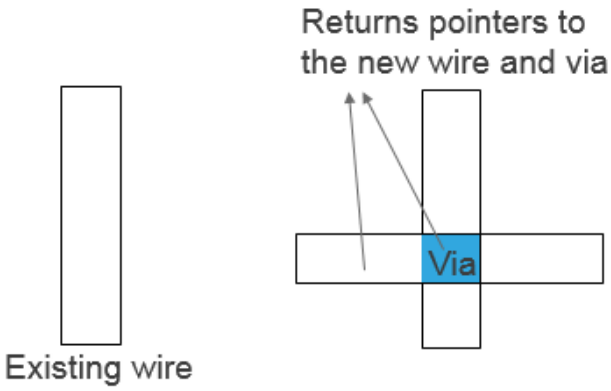
```
edit_wire_return_obj_pointer {true|false}
```

Default: `false`

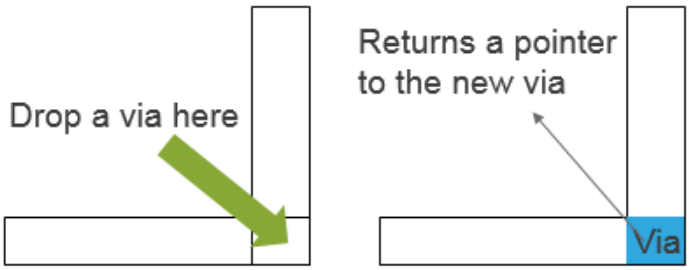
Data_type: bool, read/write

Specifies whether or not pointers are returned after wire edit operations. This attribute is set to `false` by default. When set to `true`, the wire edit commands mentioned in Related Commands return pointers for new or modified wires and vias. The returned pointers make it possible for you to select and further edit the newly created or modified wire or via. The diagrams below illustrate the pointers that are returned after some common wire edit operations:

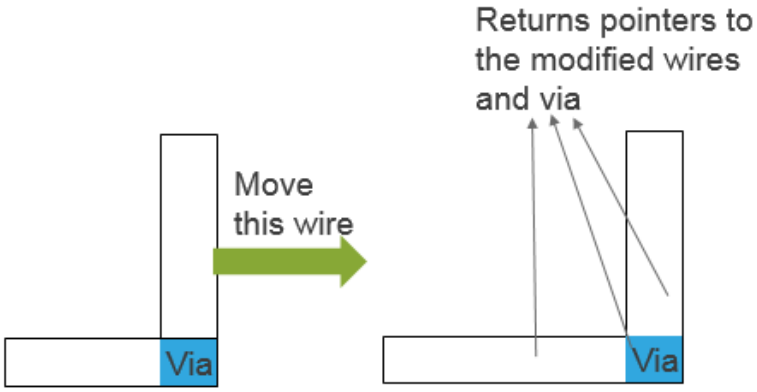
Case 1: Draw a wire

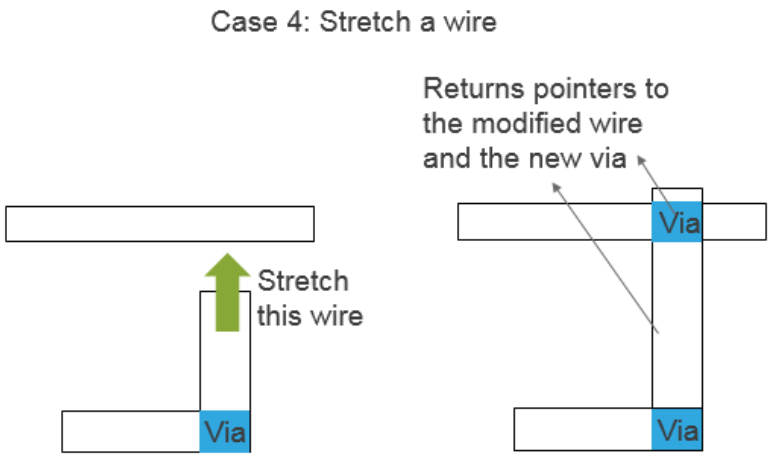


Case 2: Add a via



Case 3: Move a wire





Related Commands

The following wire edit commands return pointers for new or modified wires and vias when the `edit_wire_return_obj_pointer` attribute is set to `true`:

- `edit_create_via`
- `edit_copy`
- `edit_cut_route`
- `edit_duplicate_routes`
- `edit_end_route_point`
- `edit_merge_routes`
- `edit_move_routes`
- `edit_resize_routes`
- `edit_rotate`
- `edit_split_routes`
- `edit_trim_routes`
- `edit_update_route_layer`
- `edit_update_via`

edit_wire_rule

`edit_wire_rule` *`rule`*

Default: `default`

Data_type: string, read/write

Uses the specified LEF rule for regular wires.

Related Commands

- `delete_routes`
- `edit_add_route_point`
- `edit_merge_routes`
- `edit_move_routes`
- `edit_resize_routes`
- `edit_trim_routes`
- `edit_update_route_layer`
- `edit_update_route_net`
- `edit_update_route_rule`
- `edit_update_route_status`
- `edit_update_via`
- `select_routes`

edit_wire_search_route_rule_vias_only

```
edit_wire_search_route_rule_vias_only {true|false}
```

Default: `false`

Data_type: bool, read/write

Specifies that when the *Shift + N* bindkey is used, the Wire Editor should circle through NDR vias only.

Related Commands

- delete_routes
- edit_add_route_point
- edit_merge_routes
- edit_move_routes
- edit_resize_routes
- edit_trim_routes
- edit_update_route_layer
- edit_update_route_net
- edit_update_route_rule
- edit_update_route_status
- edit_update_via
- select_routes

edit_wire_shape

```
edit_wire_shape {ring stripe followpin iowire corewire blockwire padring blockring fillwire
fillwireopc drcfill none}
```

Default: `stripe`

Data_type: enum {ring stripe followpin iowire corewire blockwire padring blockring fillwire fillwireopc drcfill none}, read/write

Specifies the shape associated with the wire you draw.

Related Commands

- delete_routes
- edit_add_route_point
- edit_merge_routes
- edit_move_routes
- edit_resize_routes
- edit_trim_routes
- edit_update_route_layer
- edit_update_route_net
- edit_update_route_rule
- edit_update_route_status
- edit_update_via
- select_routes

edit_wire_shield

`edit_wire_shield {high|low|adjacent}`

Default: `""`

Data_type: enum* {high low adjacent}, read/write

Specifies the side to which a minimum width shield wire should be added. Use the following options to specify the side:

- `adjacent` - Specifies that a minimum width shield wire(s) should be added centered between two adjacent wires.
- `high` - Specifies that minimum width shield wires should be added at the top (High) side of horizontal routes or at the right side of vertical routes. The spacing of the added shield wire is determined by the value of the `edit_wire_outer_shield_spacing` attribute. If this value is `0`, the spacing is determined by the minimum spacing rule in the technology file.
- `low` - Specifies that minimum width shield wires should be added at the bottom (Low) side of horizontal routes or at the left side of vertical routes. The spacing of the added shield wire is determined by the value of the `edit_wire_outer_shield_spacing` attribute. If this value is `0`, the spacing is determined by the minimum spacing rule in the technology file.

Note: If you do not want to specify the side to which the minimum width shield wire should be added, set `edit_wire_shield` to `""`.

Related Commands

- delete_routes
- edit_add_route_point
- edit_merge_routes
- edit_move_routes
- edit_resize_routes
- edit_trim_routes
- edit_update_route_layer
- edit_update_route_net
- edit_update_route_rule
- edit_update_route_status
- edit_update_via
- select_routes

edit_wire_shield_look_down_layers

`edit_wire_shield_look_down_layers` *integer*

Default: `100`

Data_type: int, read/write

Specifies the number of layers below the current layer that an added shield wire can connect using a via.

Related Commands

- delete_routes
- edit_add_route_point
- edit_merge_routes
- edit_move_routes
- edit_resize_routes
- edit_trim_routes
- edit_update_route_layer
- edit_update_route_net
- edit_update_route_rule
- edit_update_route_status
- edit_update_via
- select_routes

edit_wire_shield_look_up_layers

`edit_wire_shield_look_up_layers` *integer*

Default: `100`

Data_type: int, read/write

Specifies the number of layers above the current layer that an added shield wire can connect using a via.

Related Commands

- delete_routes
- edit_add_route_point
- edit_merge_routes
- edit_move_routes
- edit_resize_routes
- edit_trim_routes
- edit_update_route_layer
- edit_update_route_net
- edit_update_route_rule
- edit_update_route_status
- edit_update_via
- select_routes

edit_wire_shield_shape

`edit_wire_shield_shape` {ring stripe followpin iowire corewire blockwire padring blockring fillwire drcfill none}

Default: `none`

Data_type: enum {ring stripe followpin iowire corewire blockwire padring blockring fillwire drcfill none}, read/write

Specifies the shape associated with the shield wire you draw.

Related Commands

- delete_routes
- edit_add_route_point
- edit_merge_routes
- edit_move_routes
- edit_resize_routes
- edit_trim_routes
- edit_update_route_layer
- edit_update_route_net
- edit_update_route_rule
- edit_update_route_status
- edit_update_via
- select_routes

edit_wire_shielding_nets

`edit_wire_shielding_nets` *nets*

Default: ""

Data_type: string, read/write

Specifies net names for shield wires, usually power or ground names.

Related Commands

- delete_routes
- edit_add_route_point
- edit_merge_routes
- edit_move_routes
- edit_resize_routes
- edit_trim_routes
- edit_update_route_layer
- edit_update_route_net
- edit_update_route_rule
- edit_update_route_status
- edit_update_via
- select_routes

edit_wire_show_drc_info_for_edit_shape

```
edit_wire_show_drc_info_for_edit_shape {true|false}
```

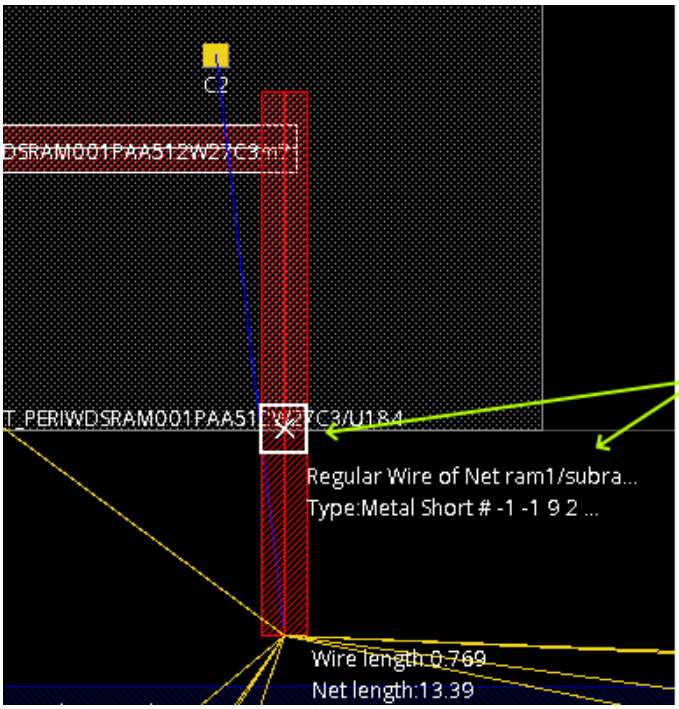
Default: `true`

Data_type: bool, read/write

Specifies that the tool should display the detailed DRC information in a violation marker nearby if a violation is generated during the following wire/via edit operations:

- Add
- Resize
- Move

In the example below, the wire being added will cause a Metal Short violation as mentioned in the DRC information displayed near the violation marker:



You can address the DRC issues displayed by either:

- Cancelling the edit operation causing the DRC violation.
- Adjusting the edit operation to meet the DRC requirement. For instance, in the example shown below, the wire being added is causing a Parallel Run Length spacing violation:

  In this case, you can move the right edge to save spacing. The DRC violation and related information will disappear:

Note: If there are any existing violation markers, they are dimmed out so that the latest marker is more clearly visible.

Related Commands

- delete_routes
- edit_add_route_point
- edit_merge_routes
- edit_move_routes
- edit_resize_routes
- edit_trim_routes
- edit_update_route_layer
- edit_update_route_net
- edit_update_route_rule
- edit_update_route_status
- edit_update_via
- select_routes

edit_wire_sibling_look_down_layers

`edit_wire_sibling_look_down_layers` *integer*

Default: `1`

Data_type: int, read/write

Specifies the number of layers below the sibling layers that an added shield wire can connect using a via.

Related Commands

- delete_routes
- edit_add_route_point
- edit_merge_routes
- edit_move_routes
- edit_resize_routes
- edit_trim_routes
- edit_update_route_layer
- edit_update_route_net
- edit_update_route_rule
- edit_update_route_status
- edit_update_via
- select_routes

edit_wire_sibling_look_up_layers

`edit_wire_sibling_look_up_layers` *integer*

Default: `1`

Data_type: int, read/write

Specifies the number of layers above the sibling layers that an added shield wire can connect using a via.

Related Commands

- delete_routes
- edit_add_route_point
- edit_merge_routes
- edit_move_routes
- edit_resize_routes
- edit_trim_routes
- edit_update_route_layer
- edit_update_route_net
- edit_update_route_rule
- edit_update_route_status
- edit_update_via
- select_routes

edit_wire_snap

```
edit_wire_snap {true | false}
```

Default: `true`

Data_type: bool, read/write

Specifies whether or not to enable wire and via snap.

Related Commands

- delete_routes
- edit_add_route_point
- edit_merge_routes
- edit_move_routes
- edit_resize_routes
- edit_trim_routes
- edit_update_route_layer
- edit_update_route_net
- edit_update_route_rule
- edit_update_route_status
- edit_update_via
- select_routes

edit_wire_snap_align_to

```
edit_wire_snap_align_to {center low high auto}
```

Default: `auto`

Data_type: enum {center low high auto}, read/write

Specifies how to align with a pin when ending a route. To ensure the proper alignment, do not specify `edit_wire_snap_objects_to_track` with this attribute. Select one of the following:

- `center`: Snaps wires to the center of the pin.
- `low`: Snaps wires to the bottom of vertical pins or to the left side of horizontal pins.
- `high`: Snaps wires to the top of vertical pins or to the right side of horizontal pins.
- `auto`: The software determines alignment automatically.

Related Commands

- delete_routes
- edit_add_route_point
- edit_merge_routes
- edit_move_routes
- edit_resize_routes
- edit_trim_routes
- edit_update_route_layer
- edit_update_route_net
- edit_update_route_rule
- edit_update_route_status
- edit_update_via
- select_routes

edit_wire_snap_bus_to_pin

```
edit_wire_snap_bus_to_pin {true|false}
```

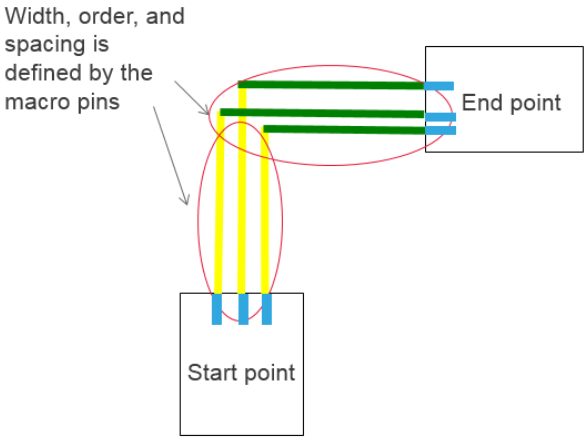
Default: `false`, which means the tool does not snap each wire of the bus to the macro pin automatically.

Data_type: bool, read/write

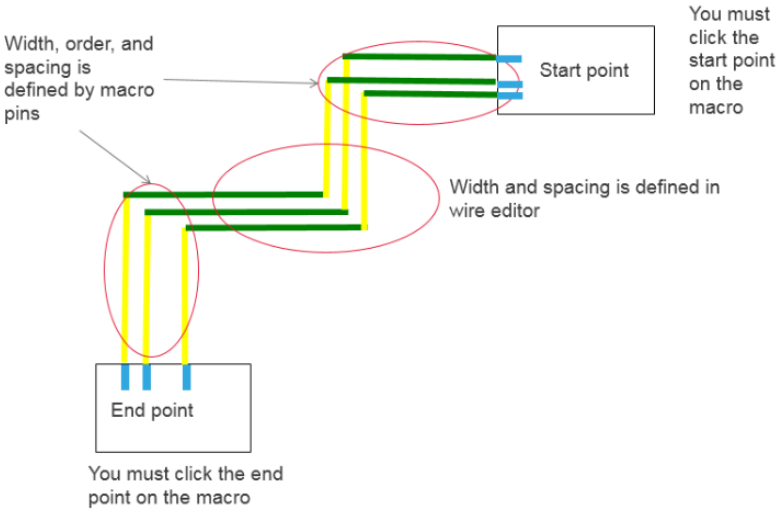
Specifies whether or not to change wire width, spacing, and order according to the connected pin.
The `edit_wire_snap_bus_to_pin` attribute supports macro to IO pin snapping as well as IO pin to IO pin snapping.

When you set the `edit_wire_snap_bus_to_pin` option to `true`:

- You do not have to align the pins at the start/end macro. Just start drawing the bus from the start macro and draw the subsequent wire segments until you arrive at the end macro. When you double-click to end the bus, all the start and end segments of the bus snap to their respective pins automatically.
- The tool gets the pin width and spacing information, and changes the wire width, spacing, and order according to the connected pins for the start and end segments of the bus.

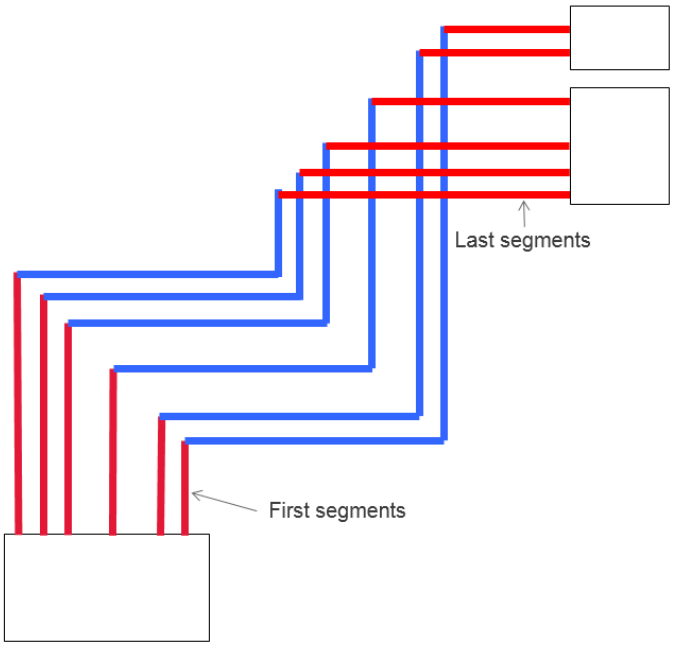


- If the bus has multiple corners, the tool uses the width and spacing value defined by the wire edit options for the middle segments of the bus.



- The tool also changes the wire segment layer according to the pin layer when accessing the pins. If the connecting pin is on a single layer, such as M4, the tool changes the wire segment layer to M4 when accessing pins. If the connecting pin is on multiple layers, the tool changes layers as follows:
    - If the pin is on M2, M3, and M4, but you have defined M5 and M6 for wire edit, the tool uses M4 for the first wire segment. In that way, wire edit has minimum via when connecting the first and second wire segment.
    - If the pin is on M2, M3, and M4, but you have defined M2 and M3 for wire edit, tool uses M3 for the first wire segment when connecting to the pin in the vertical direction (Wire edit sets M2 as the horizontal layer and M3 as the vertical layer).

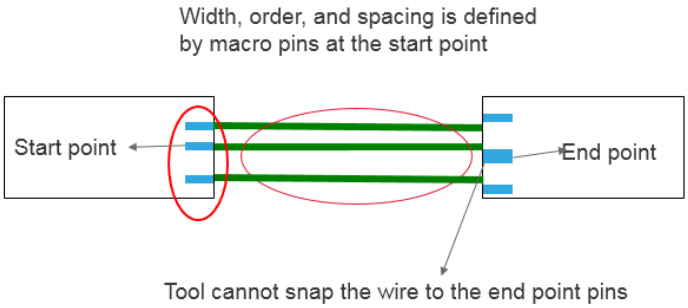
  For middle segments of the bus, the tool still uses the wire edit layer setting.
- `snap_bus_to_pin` supports snapping from one to multiple (1->N) macros for bus. When drawing the bus, the end points could be the pins on different macros. As long as the end point is inside a macro, the tool is able to auto snap to these IO cell pins. The following diagram depicts how `edit_wire_snap_bus_to_pin`, when enabled, supports 1->N macros.



- When you click outside the design core area after drawing a bus, the wires of the bus are snapped to the IO pin.

When using the `edit_wire_snap_bus_to_pin` option, keep the following limitations in mind:

- The bus routing pattern must have at least one corner. Otherwise, the tool cannot snap the bus to both the start and end target pins.



- You must click at the macros with target pins for the start and end points of the bus. The tool searches only the specified macros for the connected pins.
- For a regular wire, the tool creates an NDR for each pin width automatically (the width is not defined in tech LEF), but you cannot delete the NDR from the database.

Both regular and special wires support this feature.

Note: The `edit_wire_snap_bus_to_pin` option is exclusive to the `set_db edit_wire_snap_to pin` option. When `set_db edit_wire_snap_to pin` is set, the tool uses the original snapping behavior in which only the first drawing wire is snapped to the pin without any wire width and spacing change.

Related Commands

- delete_routes
- edit_add_route_point
- edit_merge_routes
- edit_move_routes
- edit_resize_routes
- edit_trim_routes
- edit_update_route_layer
- edit_update_route_net
- edit_update_route_rule
- edit_update_route_status
- edit_update_via
- select_routes

edit_wire_snap_end_to

```
edit_wire_snap_end_to {track_regular|track_special|manufacturing_grid}
```

Default: ""

Data_type: enum* {track_regular track_special manufacturing_grid}, read/write

Snaps the end of the wire being drawn to the specified object. You can specify the following options:

- `manufacturing_grid` - Specifies that the end of an off-grid wire should be snapped to the manufacturing grid.
- `track_regular` - Snaps the end of regular wires to track. For a long multi-pin net, you might want to draw the trunk wire first and then draw the branch wires along the trunk wire. The start/end points of the trunk wire are usually not on pins. So if the trunk wire start/end points do not snap to the track, it will leave an off-grid wire segment. To prevent this, snap the end of regular wires to track by setting `set_db edit_wire_snap_end_to track_regular`. The start/end points of the regular wire will snap to the closest orthogonal layer preferred track.
- `track_special` - Snaps the end of special routes to track. For a long multi-pin net, you might want to draw the trunk wire first and then draw the branch wires along the trunk wire. The start/end points of the trunk wire are usually not on pins. So if the trunk wire start/end points do not snap to the track, it will leave an off-grid wire segment. To prevent this, snap the end of special routes to track by setting `set_db edit_wire_snap_end_to track_special`. The start/end points of the special route will snap to the closest orthogonal layer preferred track.

Note: If you do not want to snap the end of the wire to any object, use `set_db edit_wire_snap_end_to ""`.

Related Commands

- delete_routes
- edit_add_route_point
- edit_merge_routes
- edit_move_routes
- edit_resize_routes
- edit_trim_routes
- edit_update_route_layer
- edit_update_route_net
- edit_update_route_rule
- edit_update_route_status
- edit_update_via
- select_routes

edit_wire_snap_objects_to_track

```
edit_wire_snap_objects_to_track {special|regular|patch|pin}
```

Default: `regular patch`

Data_type: enum* {special regular patch pin}, read/write

Snaps the center line of the wires being added or moved to the closest routing track in the preferred direction for the layer automatically. You can specify the following options:

- `special` - Snaps added or moved special wires to the closest routing track in the preferred direction for the layer automatically. This parameter also controls whether to cut special wires on the routing track.
  Note: See `edit_wire_snap_to pin` for a description of the software's behavior when both `edit_wire_snap_objects_to_track special` and `edit_wire_snap_to pin` are specified.
- `regular` - Snaps added or moved signal wires to the closest routing track in the preferred direction for the layer automatically. This parameter also controls whether to cut signal wires on the routing track.
  Note: See `edit_wire_snap_to pin` for a description of the software's behavior when both `edit_wire_snap_objects_to_track regular` and `edit_wire_snap_to pin` are specified.
- `patch` - Snaps patch wires automatically to the closest routing track for the layer as the wires are drawn, copied, or moved.
- `pin` - Snaps a moved physical pin to the closest routing track in the preferred direction for the layer automatically.

Note: If you do not want to snap any object to track, use `set_db edit_wire_snap_objects_to_track ""` to disable all track snap.

Related Commands

- `delete_routes`
- `edit_add_route_point`
- `edit_merge_routes`
- `edit_move_routes`
- `edit_resize_routes`
- `edit_trim_routes`
- `edit_update_route_layer`
- `edit_update_route_net`
- `edit_update_route_rule`
- `edit_update_route_status`
- `edit_update_via`
- `select_routes`

edit_wire_snap_to

`edit_wire_snap_to {special|pg|pin|row}`

Default: `special pg pin`

Data_type: enum* {special pg pin row}, read/write

Snaps the end of the wires being added or moved to the specified object. You can specify the following objects:

- `special` - Automatically extends a special wire for maximum via coverage when crossing over a same-net wire on an adjacent layer.
- `pg` - Snaps new wires whose start point is within a "snap region" to the closest end of a power or ground wire on the same net. The snap region is one-half the width of the target wire.
  - The snap region for orthogonal wires forms a square with the width of the power or ground wire.
  - The snap region for wires at 45-degree angles to the power or ground wire forms a square from the point where the wire will connect. Each side of the square is the width of the power or ground wire.
  - The snap region for wires that are parallel to power or ground wires is the full power or ground wire.
- `pin` - Snaps added or moved wires to a pin automatically. If you double-click at a pin, it snaps the wire to the pin. To ensure the wire is snapped to the center of the pin, specify `edit_wire_snap_objects_to_track`

`special` (or `edit_wire_snap_objects_to_track regular`) and specify a value for `edit_wire_snap_align_to`.

When you specify both `edit_wire_snap_objects_to_track special` (or `edit_wire_snap_objects_to_track regular`) and `edit_wire_snap_to pin`, the software has the following behavior:

- If the wire will fall partially outside the pin when snapped to a track, it snaps to the pin.
- If the wire will fall inside the pin when snapped to a track, it snaps to both the pin and a track.
- If there is no track on the pin, the wire snaps to the pin.

- `row` - Snaps added or moved wires to the nearest standard cell row. Use this parameter to create followpins. This parameter supports both horizontal and vertical rows.

For example, if you want to snap a wire to a pin automatically, specify `set_db edit_wire_snap_to pin` instead.

Note: If you do not want to snap wires to any object, use `set_db edit_wire_snap_to ""` to disable all object snap.

Related Commands

- `delete_routes`
- `edit_add_route_point`
- `edit_merge_routes`
- `edit_move_routes`
- `edit_resize_routes`
- `edit_trim_routes`
- `edit_update_route_layer`
- `edit_update_route_net`
- `edit_update_route_rule`
- `edit_update_route_status`
- `edit_update_via`
- `select_routes`

edit_wire_snap_to_track_honor_color

`edit_wire_snap_to_track_honor_color {true|false}`

Default: `true`

Data_type: bool, read/write

Specifies whether to snap wires to a track with the same color automatically. This option is on by default. If you need to move a wire to a track with a different color, set this option to `false`.

Notes:

This is a global setting and can be used to control all the snapping behavior in wire edit commands.

This command has no impact on `edit_add_route_point`. When drawing wires, the tool honors the color setting specified by `set_db edit_wire_assign_multi_pattern_color`.

Related Commands

- `delete_routes`
- `edit_merge_routes`
- `edit_move_routes`
- `edit_resize_routes`
- `edit_trim_routes`
- `edit_update_route_layer`
- `edit_update_route_net`
- `edit_update_route_rule`
- `edit_update_route_status`
- `edit_update_via`
- `select_routes`

edit_wire_snap_trim_metal_to_trim_grid

`edit_wire_snap_trim_metal_to_trim_grid {true|false}`

Default: `true`

Data_type: bool, read/write

Specifies whether or not the trim metal should snap to the closest trim track during trim-metal editing.

If you set this attribute to `false`, the trim metal is drawn on the location specified by `edit_add_trim_metal -location`.

Note: `edit_move_routes` does not follow the setting specified
by the edit_wire_snap_trim_metal_to_trim_grid attribute.

Related Commands

- delete_routes
- edit_add_route_point
- edit_merge_routes
- edit_move_routes
- edit_resize_routes
- edit_trim_routes
- edit_update_route_layer
- edit_update_route_net
- edit_update_route_rule
- edit_update_route_status
- edit_update_via
- select_routes

edit_wire_spacing

`edit_wire_spacing` *`value`*

Default: `""`

Data_type: string, read/write

Specifies the distance between wires.

Related Commands

- delete_routes
- edit_add_route_point
- edit_merge_routes
- edit_move_routes
- edit_resize_routes
- edit_trim_routes
- edit_update_route_layer
- edit_update_route_net
- edit_update_route_rule
- edit_update_route_status
- edit_update_via
- select_routes

edit_wire_spacing_horizontal

`edit_wire_spacing_horizontal` *`value`*

Default: `""`

Data_type: string, read/write

Specifies the distance between horizontal wires.

Related Commands

- delete_routes
- edit_add_route_point
- edit_merge_routes
- edit_move_routes
- edit_resize_routes
- edit_trim_routes
- edit_update_route_layer
- edit_update_route_net
- edit_update_route_rule
- edit_update_route_status
- edit_update_via
- select_routes

edit_wire_spacing_vertical

`edit_wire_spacing_vertical` *value*

Default: ""

Data_type: string, read/write

Specifies the distance between vertical wires.

Related Commands

- delete_routes
- edit_add_route_point
- edit_merge_routes
- edit_move_routes
- edit_resize_routes
- edit_trim_routes
- edit_update_route_layer
- edit_update_route_net
- edit_update_route_rule
- edit_update_route_status
- edit_update_via
- select_routes

edit_wire_status

```
edit_wire_status {cover fixed noshield routed shield auto}
```

Default: `fixed`

Data_type: enum {cover fixed noshield routed shield auto}, read/write

Specifies the status associated with the wire you draw. By default, when you add special wires manually using the GUI, the tool creates crossover vias with `fixed` status. Use the `edit_wire_status` attribute to draw wires and vias with the specified status.

If the value is specified as `auto`, the tool sets the status of wires drawn on signal and power nets to `routed` and that for wires drawn on clock nets to `fixed`.

Note:

- For a special wire, `noshield` is not supported. If you specify this value for a special wire, the tool gives a warning message and keeps the status as `fixed`.
- For a regular wire, `shield` is not supported. If you specify this value for a regular wire, the tool gives a warning message and keeps the status as `fixed`.

Related Commands

- delete_routes
- edit_add_route_point
- edit_merge_routes
- edit_move_routes
- edit_resize_routes
- edit_trim_routes
- edit_update_route_layer
- edit_update_route_net
- edit_update_route_rule
- edit_update_route_status
- edit_update_via
- select_routes

edit_wire_stop_at_drc

```
edit_wire_stop_at_drc {true|false}
```

Default: `true`

Data_type: bool, read/write

Checks DRC rules and stops moving or stretching wires before causing a violation.

- Specify `true` to stop moving or stretching the wire at the closest location available before a violation occurs. In this case, you cannot move a wire to the desired destination if that wire passes through a location where a DRC violation occurs.
- Specify `false` to ignore DRC rules when moving or stretching wires. The software moves or stretches a wire to the desired location even if it results in a DRC violation.

Related Commands

- delete_routes
- edit_add_route_point
- edit_merge_routes
- edit_move_routes
- edit_resize_routes
- edit_trim_routes
- edit_update_route_layer
- edit_update_route_net
- edit_update_route_rule
- edit_update_route_status
- edit_update_via
- select_routes

edit_wire_stretch_end

```
edit_wire_stretch_end {high|low}
```

Default: `high`

Data_type: enum {high low}, read/write

Specifies the direction in which to stretch or reduce wires.

- `high` - Stretches or reduces wires from the top for vertical wires or from the right for horizontal wires.
- `low` - Stretches or reduces wires from the bottom for vertical wires or from the left for horizontal wires.

Related Commands

- delete_routes
- edit_add_route_point
- edit_merge_routes
- edit_move_routes
- edit_resize_routes
- edit_trim_routes
- edit_update_route_layer
- edit_update_route_net
- edit_update_route_rule
- edit_update_route_status
- edit_update_via
- select_routes

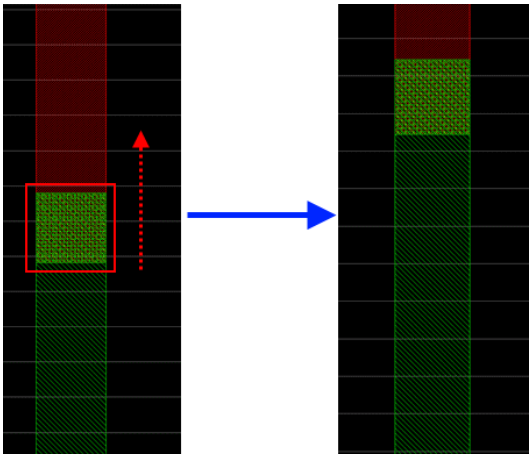
edit_wire_stretch_with_intersection

```
edit_wire_stretch_with_intersection {true|false}
```

Default: `false`

Data_type: bool, read/write

Enables wires with intersects to be stretched easily. If this option is set to `true`, you can select the via and the edge of metals and then stretch the wires and via together in a single step.



Related Commands

- delete_routes
- edit_add_route_point
- edit_merge_routes
- edit_move_routes
- edit_resize_routes
- edit_trim_routes
- edit_update_route_layer
- edit_update_route_net
- edit_update_route_rule
- edit_update_route_status
- edit_update_via
- select_routes

edit_wire_sub_class

`edit_wire_sub_class {`*`subclass`*`}`

Default: `""`

Data_type: string, read/write

Specifies the subclass name.

Related Commands

- delete_routes
- edit_add_route_point
- edit_merge_routes
- edit_move_routes
- edit_resize_routes
- edit_trim_routes
- edit_update_route_layer
- edit_update_route_net
- edit_update_route_rule
- edit_update_route_status
- edit_update_via
- select_routes

edit_wire_turn_at

```
edit_wire_turn_at {center_line|wire_edge}
```

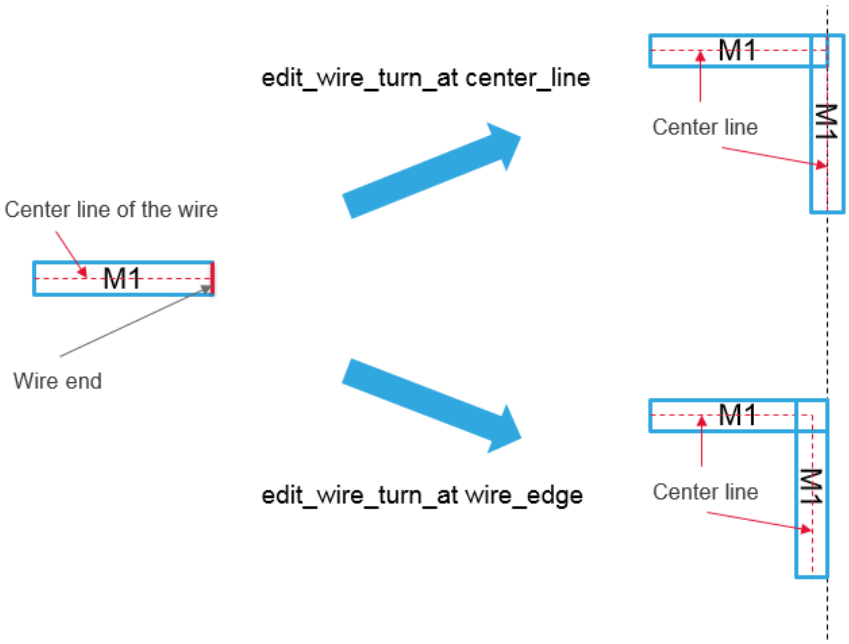
Default: `center_line`

Data_type: enum* {center_line wire_edge}, read/write

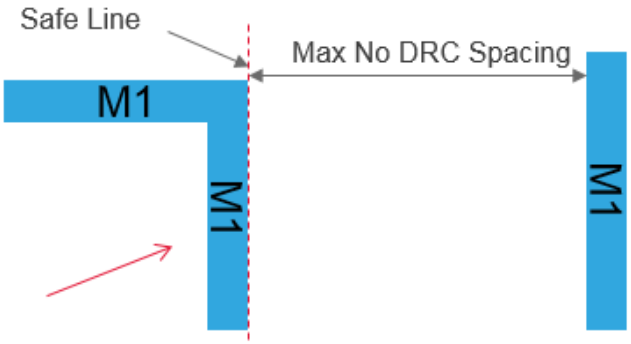
Specifies the start location of the new wire segment when a wire is turned. This attribute can be used for both regular and special wires. You can specify one of the following values for `edit_wire_turn_at`:

- `center_line` - When you turn a wire, the center line of the new wire segment aligns to the end of the old wire segment.
- `wire_edge` - When you turn a wire, the long edge of the new wire segment aligns to the end of the old wire segment.

The diagram below depicts the tool behavior for both options:



Use the `wire_edge` option when turning a wire at the *Safe Line* for an adjacent wire. With this option, the tool turns the wire without extending it beyond the *Safe Line* to avoid a minumum spacing rule violation with the adjacent wire:



You can use the default `edit_wire_turn_at center_line` option if there is no adjacent wire.

Related Commands

- delete_routes
- edit_add_route_point
- edit_merge_routes
- edit_move_routes
- edit_resize_routes
- edit_trim_routes
- edit_update_route_layer
- edit_update_route_net
- edit_update_route_rule
- edit_update_route_status
- edit_update_via
- select_routes

edit_wire_type

```
edit_wire_type {regular|special|patch}
```

Default: ""

Data_type: enum {regular special patch}, read/write

Specifies the type of the wire being created. You can specify one of the following options:

- `regular` - Creates regular wires and vias.
- `special` - Creates special wires and vias.
- `patch` - Creates patch wires.

Related Commands

- delete_routes
- edit_add_route_point
- edit_merge_routes
- edit_move_routes
- edit_resize_routes
- edit_trim_routes
- edit_update_route_layer
- edit_update_route_net
- edit_update_route_rule
- edit_update_route_status
- edit_update_via
- select_routes

edit_wire_unrestricted_regular_wire_width

```
edit_wire_unrestricted_regular_wire_width {true|false}
```

Default: `false`

Data_type: bool, read/write

Specifies whether the widening of a regular wire is unrestricted. By default, when you are changing the width of a wire, it snaps to the closest non-default route (NDR) available. If you set the `edit_wire_unrestricted_regular_wire_width` parameter to `true`, a wire can be stretched to any width without any restrictions.

Related Commands

- delete_routes
- edit_add_route_point
- edit_merge_routes
- edit_move_routes
- edit_resize_routes
- edit_trim_routes
- edit_update_route_layer
- edit_update_route_net
- edit_update_route_rule
- edit_update_route_status
- edit_update_via
- select_routes

edit_wire_update_shield_net

```
edit_wire_update_shield_net {true|false}
```

Default: `false`

Data_type: bool, read/write

Specifies whether the shield net attribute on a signal net should be automatically updated. Use this attribute if you want Nanoroute to recognize the net as a shield net and not remove it during routing.

If you set the `edit_wire_update_shield_net` attribute to `true` before drawing a signal net, for example `net11`, the shield net attribute is automatically set on that signal net. You can verify this by using the `get_route_attributes` command as follows:

```
> get_route_attributes -nets net11
```

The result is:

```
#NET net11 attributes:
# weight = 2 (default)
# non_default_rule = default (default)
# non_default_rule_effort = auto (default)
# shield_net = VDD
```

If you now run `route_design`, NanoRoute recognizes the net as a shield net and does not remove it.

Related Commands

- delete_routes
- edit_add_route_point
- edit_merge_routes
- edit_move_routes
- edit_resize_routes
- edit_trim_routes
- edit_update_route_layer
- edit_update_route_net
- edit_update_route_rule
- edit_update_route_status
- edit_update_via
- select_routes

edit_wire_use_fix_via

```
edit_wire_use_fix_via {true|false}
```

Default: `false`

Data_type: bool, read/write

Specifies whether to use fixVia for special via with minStep DRC violation.

Note: If a design is under 20nm, the tool turns on fixVia integration by default:
`set_db edit_wire_use_fix_via true`

To turn off the setting, use `set_db edit_wire_use_fix_via false`.

Related Commands

- delete_routes
- edit_add_route_point
- edit_merge_routes
- edit_move_routes
- edit_resize_routes
- edit_trim_routes
- edit_update_route_layer
- edit_update_route_net
- edit_update_route_rule
- edit_update_route_status
- edit_update_via
- select_routes

edit_wire_use_interleaving_wire_group

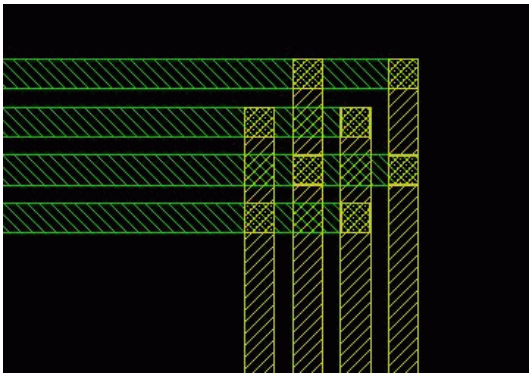
`edit_wire_use_interleaving_wire_group {true|false}`

Default: `false`

Data_type: bool, read/write

Alternates the specified wires in the wire group. For example, if you specify `set_db edit_wire_nets VDD VSS` and `set_db edit_wire_use_interleaving_wire_group true`, the software alternates the power and ground wires in the wire group.

When `edit_wire_use_interleaving_wire_group` is set to `true`, the software draws a wire group like the following:



Related Commands

- delete_routes
- edit_add_route_point
- edit_merge_routes
- edit_move_routes
- edit_resize_routes
- edit_trim_routes
- edit_update_route_layer
- edit_update_route_net
- edit_update_route_rule
- edit_update_route_status
- edit_update_via
- select_routes

edit_wire_use_wire_group

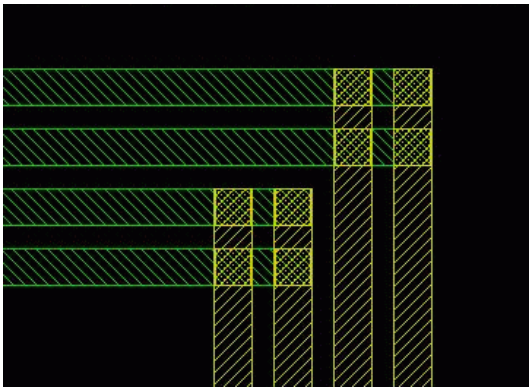
```
edit_wire_use_wire_group {true|false}
```

Default: `false` (The wire editor does not use wire groups.)

Data_type: bool, read/write

Groups multiple wires from the same net. Specify this attribute to decrease resistance and provide stronger connections.

The following figure shows power and ground wires that form a wire group.



Related Commands

- delete_routes
- edit_add_route_point
- edit_merge_routes
- edit_move_routes
- edit_resize_routes
- edit_trim_routes
- edit_update_route_layer
- edit_update_route_net
- edit_update_route_rule
- edit_update_route_status
- edit_update_via
- select_routes

edit_wire_use_wire_group_bits

`edit_wire_use_wire_group_bits` *`number_of_bits`*

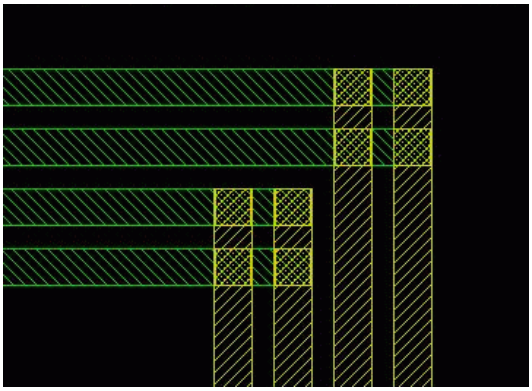
Default: `0` (disabled)

Data_type: int, read/write

Specifies the number of times to replicate a wire in the wire group. For example, if you specify `VSS` and `VDD` nets, and specify `2` for this attribute, the software creates a wire group with two power wires and two ground wires.

Note - The value of this attribute must be greater than `1` to enable wire groups.

When `edit_wire_use_wire_group_bits 2` is specified, Innovus draws a wire group like the following:



Related Commands

- delete_routes
- edit_add_route_point
- edit_merge_routes
- edit_move_routes
- edit_resize_routes
- edit_trim_routes
- edit_update_route_layer
- edit_update_route_net
- edit_update_route_rule
- edit_update_route_status
- edit_update_via
- select_routes

edit_wire_use_wire_group_reinforcement

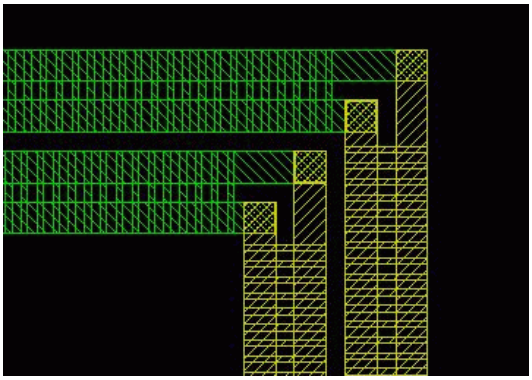
```
edit_wire_use_wire_group_reinforcement {true|false}
```

Default: `false`

Data_type: bool, read/write

Creates slots in the wire group by creating wires that connect the wires in the wire group, but are orthogonal to them. Slots help to reduce resistance in power networks. Specifying both this attribute and `edit_wire_use_interleaving_wire_group` creates shorts in the wire group.

When `edit_wire_use_wire_group_reinforcement true` is specified, Innovus draws a wire group like the following:



Related Commands

- delete_routes
- edit_add_route_point
- edit_merge_routes
- edit_move_routes
- edit_resize_routes
- edit_trim_routes
- edit_update_route_layer
- edit_update_route_net
- edit_update_route_rule
- edit_update_route_status
- edit_update_via
- select_routes

edit_wire_use_wire_group_reinforcement_group_via

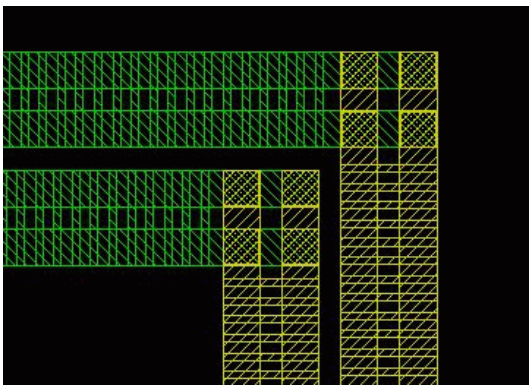
```
edit_wire_use_wire_group_reinforcement_group_via {true|false}
```

Default: `false`

Data_type: bool, read/write

Specifies that the wire group uses group vias. Group vias are connected vias in a wire group. Using group vias ensures that the contact is maintained if one of the vias in the wire group produces a void.

In the following figure, the wire group has group vias:



Related Commands

- delete_routes
- edit_add_route_point
- edit_merge_routes
- edit_move_routes
- edit_resize_routes
- edit_trim_routes
- edit_update_route_layer
- edit_update_route_net
- edit_update_route_rule
- edit_update_route_status
- edit_update_via
- select_routes

edit_wire_use_wire_group_reinforcement_spacing

```
edit_wire_use_wire_group_reinforcement_spacing spacing
```

Default: `0.0`

Data_type: double, read/write

Specifies the spacing for the orthogonal wires in the wire groups.

Related Commands

- delete_routes
- edit_add_route_point
- edit_merge_routes
- edit_move_routes
- edit_resize_routes
- edit_trim_routes
- edit_update_route_layer
- edit_update_route_net
- edit_update_route_rule
- edit_update_route_status
- edit_update_via
- select_routes

edit_wire_use_wire_group_reinforcement_width

`edit_wire_use_wire_group_reinforcement_width` *spacing*

Default: `0.0`

Data_type: double, read/write

Specifies the width of the orthogonal wires in the wire groups.

Related Commands

- delete_routes
- edit_add_route_point
- edit_merge_routes
- edit_move_routes
- edit_resize_routes
- edit_trim_routes
- edit_update_route_layer
- edit_update_route_net
- edit_update_route_rule
- edit_update_route_status
- edit_update_via
- select_routes

edit_wire_verbose

```
edit_wire_verbose {true|false}
```

Default: `false`

Data_type: bool, read/write

Reports untrimmed wires/vias that cannot be trimmed because either they are connected to a patch wire or their status is `cover`. Set this parameter to `true` before running `edit_trim_routes` if you want untrimmed wires/vias to be reported. You can additionally use `edit_wire_debug_file` to specify the name for the report. The default report file name is `editTrim_debug.rpt`.

For example, the following set of commands report untrimmed wires and vias with connectivity issues in a file named `wireTest.rpt`:

```
set_db edit_wire_verbose true
set_db edit_wire_debug_file wireTest.rpt
edit_trim_routes
```

The report (sample) is as follows:

```
############################################################
# Generated by: Cadence Innovus 20.10-b015_1
# OS: Linux x86_64(Host ID noi-leenap1)
# Generated on: Mon Feb 10 13:50:19 2020
# Design: carve
# Command: edit_trim_routes
############################################################
```

```
Removed 0 wires and 0 vias.
1 wire can't be trimmed, because it connect a patch wire or the status is Cover.

The wires status is Cover:
Wire: ram1/subram02_lower/TESTACT_INST_PERIWDSRAM001PAA512W27C3/n7 M2 97845,-642213,100551,-642147
```

Note: Only wires and vias that have connectivity issues are reported.

Related Commands

- edit_trim_routes
- edit_update_route_status

edit_wire_via_columns

edit_wire_via_columns *integer*

Default: 0

Data_type: int, read/write

Specifies the number of columns in a via cut. This parameter works only if -use_via_gen is set to 1 for special vias.

Related Commands

- delete_routes
- edit_add_route_point
- edit_merge_routes
- edit_move_routes
- edit_resize_routes
- edit_trim_routes
- edit_update_route_layer
- edit_update_route_net
- edit_update_route_rule
- edit_update_route_status
- edit_update_via
- select_routes

edit_wire_via_exclude_spec

`edit_wire_via_exclude_spec` *value*

Default: `""`

Data_type: string, read/write

Specifies the list of vias to be ignored by the wire editor. You can define the list of vias in the *exclude_list* as follows:

```
set_db edit_wire_via_exclude_spec {{default via0} {rule1 via2} {rule2 {via2 via3}}}
```

For example, the wire editor will ignore the `V1L2LVH` via with the following command:

```
set_db edit_wire_via_exclude_spec {{default V1L2LVH}}
```

Related Commands

- `delete_routes`
- `edit_add_route_point`
- `edit_merge_routes`
- `edit_move_routes`
- `edit_resize_routes`
- `edit_trim_routes`
- `edit_update_route_layer`
- `edit_update_route_net`
- `edit_update_route_rule`
- `edit_update_route_status`
- `edit_update_via`
- `select_routes`

edit_wire_via_override_spec

`edit_wire_via_override_spec` *value*

Default: `""`

Data_type: string, read/write

Specifies the default vias to be used by wire editor. You can use this attribute if you do not want wire editor to select vias automatically.

`edit_wire_via_override_spec` allows a regular via on NDR nets. You can define the via in the via spec as follows:

```
set_db edit_wire_via_override_spec {{default via0} {rule1 via2} {rule2 via1}}
```

Here, if you draw a wire with NDR name `rule1`, the tool will always drop `via2` in the wire intersection area. The tool will not check whether `via2` belongs to NDR rule `rule1` in the Tech Lef.

You can define the full set of required vias for regular as well as NDR nets in one command with the `edit_wire_via_override_spec` attribute. For example:

```
set_db edit_wire_via_override_spec { {default via0 via1 via2 via3 via4 via5 via6 via7 via8} {NDR_rule1
via2 via3 via4 via5 via6 via7 NDR_rule1_via2 NDR_rule1_via3 NDR_rule1_via4 NDR_rule1_via5
NDR_rule1_via6 NDR_rule1_via7}  {NDR_rule2 via2 via3 via4 via5 via6 via7 NDR_rule1_via2 NDR_rule1_via3
NDR_rule1_via4 NDR_rule1_via5 NDR_rule2_via2 NDR_rule2_via3 NDR_rule2_via4 NDR_rule1_via5
NDR_rule2_via6 NDR_rule2_via7} }
```

This use model reduces the runtime for via selection during Wire Edit.

When you perform wire edit operations, via selection happens as follows for the above definition:

- For regular wire edit, the tool always picks a via from the `default` list specified with the `edit_wire_via_override_spec` option.
- For editing nets with the `NDR_rule1` rule, the tool picks any via from the list defined under the rule `NDR_rule1` in the above example.
- For editing nets with `NDR_rule2`, the tool picks any via from the list defined under the rule `NDR_rule2`.

During wire edit for DRC fixing on both regular and NDR wires, the tool always picks a via that is DRC clean after any wire edit operation, from the above definition.

The following command uses the `edit_wire_via_override_spec` attribute to specify via selections for the default rule as well as for the nondefault rule `SP`:

```
set_db edit_wire_via_override_spec {{default via12 via23 via45} {SP spvia12}}
```

As specified in the command, wire editor makes via selections for layers as follows:

- For the default rule:
  - `via12` for layer `V12`
  - `via23` for layer `V23`
  - `via45` for layer `V45`
- For the nondefault rule `SP`:
  - `spvia12` for layer `V12`

Note: Wire editor automatically selects best vias for the layers that are not specified with `edit_wire_via_override_spec`.

Related Commands

- delete_routes
- edit_add_route_point
- edit_merge_routes
- edit_move_routes
- edit_resize_routes
- edit_trim_routes
- edit_update_route_layer
- edit_update_route_net
- edit_update_route_rule
- edit_update_route_status
- edit_update_via
- select_routes

edit_wire_via_rows

`edit_wire_via_rows` *integer*

Default: `0`

Data_type: int, read/write

Specifies the number of rows in a via cut. This parameter works only if `-use_via_gen` is set to `1` for special vias.

Related Commands

- delete_routes
- edit_add_route_point
- edit_merge_routes
- edit_move_routes
- edit_resize_routes
- edit_trim_routes
- edit_update_route_layer
- edit_update_route_net
- edit_update_route_rule
- edit_update_route_status
- edit_update_via
- select_routes

edit_wire_via_scale_height

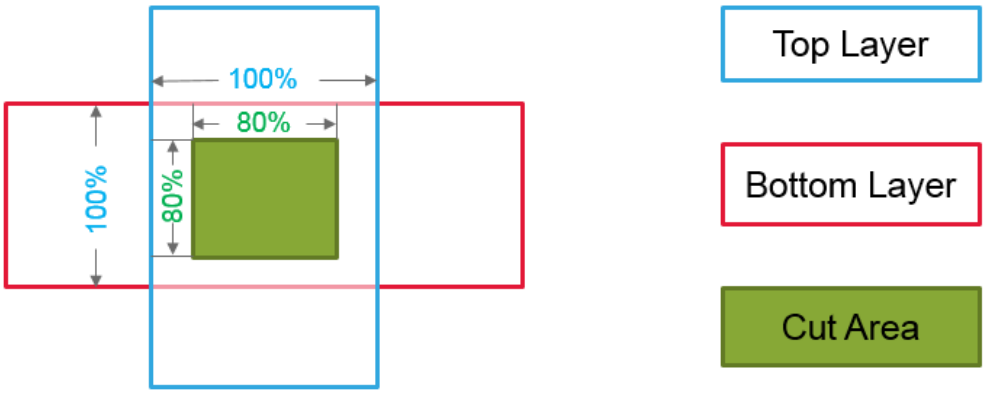
`edit_wire_via_scale_height` *value*

Default: `100`

Data_type: int, read/write

Specifies the absolute height of the special vias added when drawing wires. For example, the following commands scale the via added while drawing a wire to 80% height and width:

```
set_db edit_wire_via_scale_height 80
set_db edit_wire_via_scale_width 80
```



Related Commands

- delete_routes
- edit_add_route_point
- edit_merge_routes
- edit_move_routes
- edit_resize_routes
- edit_trim_routes
- edit_update_route_layer
- edit_update_route_net
- edit_update_route_rule
- edit_update_route_status
- edit_update_via
- select_routes

edit_wire_via_scale_width

`edit_wire_via_scale_width` *`value`*

Default: `100`

Data_type: int, read/write

Specifies the absolute width of the special vias added when drawing wires.

Related Commands

- delete_routes
- edit_add_route_point
- edit_merge_routes
- edit_move_routes
- edit_resize_routes
- edit_trim_routes
- edit_update_route_layer
- edit_update_route_net
- edit_update_route_rule
- edit_update_route_status
- edit_update_via
- select_routes

edit_wire_via_snap_to_intersection

```
edit_wire_via_snap_to_intersection {true|false}
```

Default: `true`

Data_type: bool, read/write

Specifies whether to snap vias to the intersection of wires.

Related Commands

- delete_routes
- edit_add_route_point
- edit_merge_routes
- edit_move_routes
- edit_resize_routes
- edit_trim_routes
- edit_update_route_layer
- edit_update_route_net
- edit_update_route_rule
- edit_update_route_status
- edit_update_via
- select_routes

edit_wire_width

`edit_wire_width` *`value`*

Default: `0.0`

Data_type: double, read/write

Specifies the width of patch wires.

Related Commands

- delete_routes
- edit_add_route_point
- edit_merge_routes
- edit_move_routes
- edit_resize_routes
- edit_trim_routes
- edit_update_route_layer
- edit_update_route_net
- edit_update_route_rule
- edit_update_route_status
- edit_update_via
- select_routes

edit_wire_width_horizontal

`edit_wire_width_horizontal` *value*

Default: `0.0`

Data_type: double, read/write

Specifies the width for horizontal wires.

Related Commands

- delete_routes
- edit_add_route_point
- edit_merge_routes
- edit_move_routes
- edit_resize_routes
- edit_trim_routes
- edit_update_route_layer
- edit_update_route_net
- edit_update_route_rule
- edit_update_route_status
- edit_update_via
- select_routes

edit_wire_width_vertical

`edit_wire_width_vertical` *value*

Default: `0.0`

Data_type: double, read/write

Specifies the width for vertical wires.

Related Commands

- delete_routes
- edit_add_route_point
- edit_merge_routes
- edit_move_routes
- edit_resize_routes
- edit_trim_routes
- edit_update_route_layer
- edit_update_route_net
- edit_update_route_rule
- edit_update_route_status
- edit_update_via
- select_routes

edit_wire_wire_override_spec

edit_wire_wire_override_spec {*specification*}

Default: ""

Data_type: string, read/write

Contains the details of the override specification. The specification has the following format:

{{*wire_index width spacing layer ndr_name*} {...}}

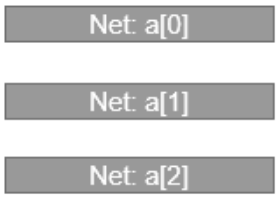
Specify the net numbers, followed by the override width, the override spacing to the next net, layer, or override NDR name for the specified nets.

The net number is the position of the net specified by the -nets parameter. If you specify more than one net, separate net numbers with commas.

For example, the following specification indicates that the first and second nets should have a width of 10.0 and a spacing of 3.0:

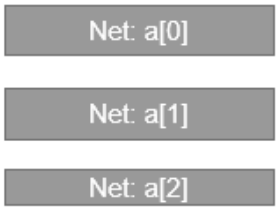
1,2 10.0 3.0

Now, consider the addition of a bus a[0] a[1] a[2]. The NDR name is default, and the metal layer name is M0.



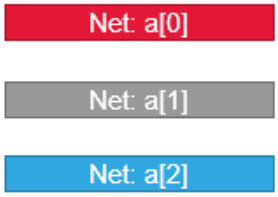
- The following command reassigns the first and second nets to a new NDR, `2_wide`:

  ```
  set_db edit_wire_wire_override_spec {1,2 "" "" "" 2_wide}
  ```

- The following command assigns the `M2` layer to the first wire and the `M4` layer to the third wire.:

  ```
  set_db edit_wire_wire_override_spec {{1 "" M2} {3 "" M4}}
  ```

Related Commands

- `delete_routes`
- `edit_add_route_point`
- `edit_merge_routes`
- `edit_move_routes`
- `edit_resize_routes`
- `edit_trim_routes`
- `edit_update_route_layer`
- `edit_update_route_net`
- `edit_update_route_rule`
- `edit_update_route_status`
- `edit_update_via`
- `select_routes`

end_parallel_edit

```
end_parallel_edit
[-help]
-out_file parallel_edit_file_name
```

Ends parallel editing. The conmmand:

- compares the data in memory with the original data saved in binary files by `start_parallel_edit`
- saves the parallel edit file with the specified name

Multiple `end_parallel_edit` commands are allowed after one `start_parallel_edit`.

Parameters

`-help`	Prints a brief description that includes type and default information for each `end_parallel_edit` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man end_parallel_edit`
`-out_file` *`parallel_edit_file_name`*	Specifies the name of the parallel edit file. *Data_type*: string, required

Example

- The following command compares the data in memory with the data saved in the binary files and writes the parallel edit information to `PE1.txt` file: :

```
innovus 8> end_parallel_edit -out_file PE1.txt
Start saving parallel edit information to PE1.txt
************************************************************************
Comparing net attribute and regular wire/via with reference DB
Comparing instance with reference DB
Comparing physical pin with reference DB
Comparing NDR rule with reference DB
Comparing routing blockage with reference DB
Comparing placement blockage with reference DB
Comparing via cell with reference DB
Comparing special wire/via with reference DB
************************************************************************
End saving parallel edit information
```

Related Information

- `read_parallel_edit_files`
- `start_parallel_edit`
- Editing Wires chapter in the *User Guide*
  - *Parallel Editing Capability*

read_parallel_edit_files

```
read_parallel_edit_files
[-help]
[-check]
[-conflict_mode {only_one stop_all}]
-files list_of_files
[-ignore_error]
[-report_file file_name]
```

Loads all the specified parallel edit files and implements the editing changes recorded in them.

During parallel editing, there is a possibility of a conflict if the same attribute of an object is changed in different parallel edit files. Some cases in which a conflict can occur are listed below:

- Two users edit the layer of the same wire segment.
- Two users edit the width of the same NDR rule.
- One user cuts the wire segment, and another user changes the status of the same wire.

In the following examples of parallel edits, there is no conflict:

- One user edits the layer of wire, and another user edits its mask.
- One user edits the width of an NDR rule, and another user edits its spacing.

In case of any conflict in the parallel edit files, the tool implements changes as per the specified conflict mode and writes the conflicting information to a report file.

For information on the parallel edit flow, format of a parallel edit file, and conflict handling examples see the "Parallel Editing Capability" section in the Editing Wires chapter of the *User Guide.*

Parameters

`-help`	Prints a brief description that includes type and default information for each `read_parallel_edit_files` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man read_parallel_edit_files`
`-check`	Specifies that the tool should only check the syntax and conflict of the files but should not actually implement the changes. *Data_type*: bool, optional
`-conflict_mode {only_one stop_all}`	Specifies the conflict mode. The two possible modes are: • `only_one` - In case of conflicting commands, the first read command is implemented and the other command is recorded in the conflict report. • `stop_all` - In case of conflicting commands, the tool implements neither of the conflicting commands and records both of them in the conflict report. *Data_type*: enum, optional
`-files` *list_of_files*	Specifies the list of parallel editing files. *Data_type*: string, required
`-ignore_error`	Specifies whether errors in files should be ignored. When this option is specified, the tool skips lines with issues in the files but continues to read the other wires on the same net. *Data_type*: bool, optional
`-report_file` *file_name*	Specifies the name of the conflict report. *Data_type*: string, optional

Related Information

- `end_parallel_edit`
- `start_parallel_edit`
- Editing Wires chapter in the *User Guide*
  - *Parallel Editing Capability*

report_legal_wire_width_for_bump

```
report_legal_wire_width_for_bump
[-help]
-location x y
-width value
```

Reports two legal widths for a 45-degree wire to connect to the 45-degree side of an octagon-shaped bump in the wire editor.

Parameters

`-help`	Prints out the command usage.
`-location x y`	Specifies the location inside the bump near the (45-degree) side of the connection. *Data_type*: point, required
`-width value`	Sets the initial width value of the 45-degree wire in microns. *Data_type*: float, required

Example

- The following command reports two legal widths for a 45-degree wire to connect to the 45-degree side of an octagon-shaped bump at the specified location with the specified initial width:

  ```
  report_legal_wire_width_for_bump -location 681.83 777.28 -width 9.25
  ```

  The reported values are:

  ```
  9.230000 9.260000
  ```

Related Information

- Editing Wires chapter in the *User Guide*
- Edit Menu chapter in the *Menu Reference*
  - *Wire* section

select_routes

```
select_routes
[-help]
[-area {x1 y1 x2 y2}]
[-direction {h | v | 45 | 135}]
[-floating_via]
[-layer list_of_layers]
[-nets net*]
[-physical_pin_only]
[-regular_wire_with_drc]
[-shapes {ring stripe followpin iowire corewire blockwire padring blockring fillwire fillwireopc
drcfill none}]
[-shield]
[-shield_only]
[-status {cover fixed noshield routed shield}]
[-sub_class sublass_name]
[-type {regular special patch}]
[-use {clock power signal}]
[-via_cell cell_name_list]
[-obj_type {wire via}]
[-with_edit_flag {0 | 1}
[-from_pin pin_name -to_pin pin_name]
```

Selects wires and vias using the specified parameters. The wires and vias are updated by subsequent wire editing commands.

Parameters

`-help`	Prints the usage of the command. For a detailed description of the command, use the `man` command: `man select_routes`
`-area {x1 y1 x2 y2}`	Specifies the coordinates for the area from which to select wires and vias. *Default:* Entire area *x1*: Specifies the lower left x coordinate. *y1*: Specifies the lower left y coordinate. *x2*: Specifies the upper right x coordinate. *y2*: Specifies the upper right y coordinate. *Data_type*: rect, optional

`-direction {h | v | 45 | 135}`	Specifies the orientation of the wires to be selected. *Default:* If you do not specify this parameter, selects wires with any orientation. `h`: Specifies that horizontal wires are to be selected. `v`: Specifies that vertical wires are to be selected. 45: Specifies that 45-degree wires are to be selected. 135: Specifies that 135-degree wires are to be selected. *Data_type*: enum, optional
`-floating_via`	Specifies that only floating vias are to be selected. A via, or a stack of vias, is floating, if it is not connected to another object, wire, pin, or via, on either its top or bottom metal layer or on both these layers. *Data_type*: bool, optional
`-from_pin` *pin_name*	Specifies the name of the starting pin of the wire to be selected. The starting pin you specify should be connected to the same net as the ending pin specified with `-to_pin`. If they connect to different nets, the tool reports an error. In addition, the from and to pins you specify should be connected to the same wire. **Note**: Selection of wires between specified pins is supported only for signal and clock nets. *Data_type*: string, optional
`-layer` *list_of_layers*	Specifies the layers from which to select wires and vias. This option supports backside and Through Silicon Via (TSV) layers as well. If you are specifying multiple layers, enclose them in braces. *Default:* If you do not specify this parameter, selects wires and vias on all layers. *Data_type*: string, optional
`-nets` *net**	Specifies the net names from which to select wires and vias. You can use wildcard values in this field. For example, if you specify `V*`, selects wires/vias from nets `VDD` and `VSS`. *Data_type*: (net)*, optional
`-obj_type {wire via}`	Specifies whether wires or vias or both should be selected. If the parameter is not specified, only wires are selected. *Default*: `wire` *Data_type*: (enum)+, optional

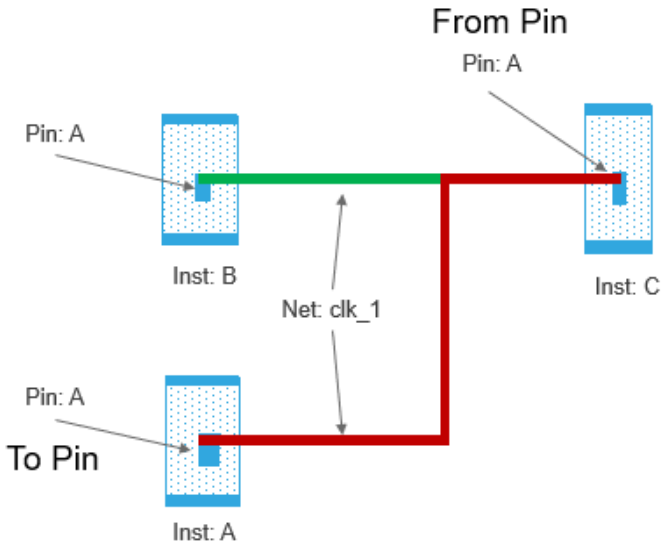
`-physical_pin_only`	Specifies that physical pins are to be selected. By default, `select_routes` does not select physical pins. You can use the `-physical_pin_only` parameter along with `-area`, `-layer`, or `-nets` to select the physical pins in the specified area, layer, or net, respectively. *Data_type*: bool, optional
`-regular_wire_with_drc`	Selects regular wires with violation markers. Clock nets are also selected. *Data_type*: bool, optional
`-shapes {ring stripe followpin iowire corewire blockwire padring blockring fillwire fillwireopc drcfill none}`	
	Specifies the shapes of the wires and vias to be selected. If you specify multiple shapes, enclose the list of shapes in braces. For shape definitions, see *Shapes,* in the "DEF Syntax" chapter of the LEF/DEF Language Reference. *Default:* If you do not specify this parameter, selects all shapes. *Data_type*: (enum)+, optional
`-shield`	Selects the wires and vias of the nets specified with the `-nets` parameter, along with the wires and vias shielding the specified nets. **Note**: This parameter cannot be used with `-shield_only`. *Data_type*: bool, optional
`-shield_only`	Selects only the shield wires associated with the nets specified using the `-nets` parameter. The signal nets being shielded are not selected. **Note**: This parameter cannot be used with `-shield`. *Data_type*: bool, optional
`-status {cover fixed noshield routed shield unknown}`	
	Specifies the status of the wires and vias to be selected. If you specify more than one status, enclose the status list in curly braces. **Note:** If you select `-type special -status noshield` or `-type signal -status shield`, the software generates an error message. *Data_type*: (enum)+, optional

`-sub_class` *`sublass_name`*	Specifies the subclass name of the wires and vias to be selected. Use this parameter if you want to select wires or vias belonging to a specific subclass. Here, *`subclass_name`* is a user-defined attribute that has been previously defined using the `-user_class` parameter of the `add_split_power_vias`, `add_rings`, `add_stripes`, or `route_special` command. For wires, use this parameter if you want to select or delete rings, stripes, or other wires belonging to a specific subclass. Only special wire is supported by this option. **Note:** You can use wildcards in *`subclass_name`*. *Data_type*: string, optional
`-to_pin` *`pin_name`*	Specifies the name of the ending pin of the wire to be selected. The ending pin you specify should be connected to the same net as the starting pin specified with `-from_pin`. If they connect to different nets, the tool reports an error. In addition, the from and to pins you specify should be connected to the same wire. If there there are more than one wire or via between the specified starting and ending pins, the command is not able to select the wire path and returns an error. The `-from_pin` and `-to_pin` parameters are compatible with only the `-direction`, `-layer`, `-object_type`, and `-status` options of the `select_routes` command. If you use `-from_pin` and `-to_pin` with any other `select_routes` parameters, the tool returns an error. **Note**: Selection of wires between specified pins is supported only for signal and clock nets. *Data_type*: string, optional
`-type {regular special patch}`	
	Specifies whether to select regular wires and vias, special wires and vias, or patch wires. **Note:** If you specify `regular`, you must omit the `-shapes` parameter or specify `-shapes none`. *Data_type*: enum, optional
`-use {clock power signal}`	Specifies whether to select only CLOCK wires/vias, POWER wires, or SIGNAL wires. *Data_type*: enum, optional
`-via_cell` *`cell_name_list`*	Specifies the via cells of the vias to be selected. You can use wildcards while specifying via cell names. If you are specifying multiple cell names, enclose them in braces. *Data_type*: string, optional

`-with_edit_flag {0 | 1}`	Selects all modified wires and vias, that is wires and vias with value `1` for the `is_edit` property on the net. **Notes** • As the `is_edit` property is added on the whole net, all the wires and vias on the net with `is_edit` value set to `1` will be selected. • The `is_edit` property is used to flag nets with modified wires and vias. This is useful when evaluating how many objects need to be modified for fixing DRC violations and then deciding whether to fix the violations manually or through ECO routing. The `is_edit` property is controlled through the `edit_wire create_is_edit_flag` root attribute in the `edit_wire` category. By default, this parameter is set to `false`. If you set the value of this attribute to `true`, the `is_edit` property is set to `1` for nets with modified wires or vias. *Data_type*: enum, optional

Examples

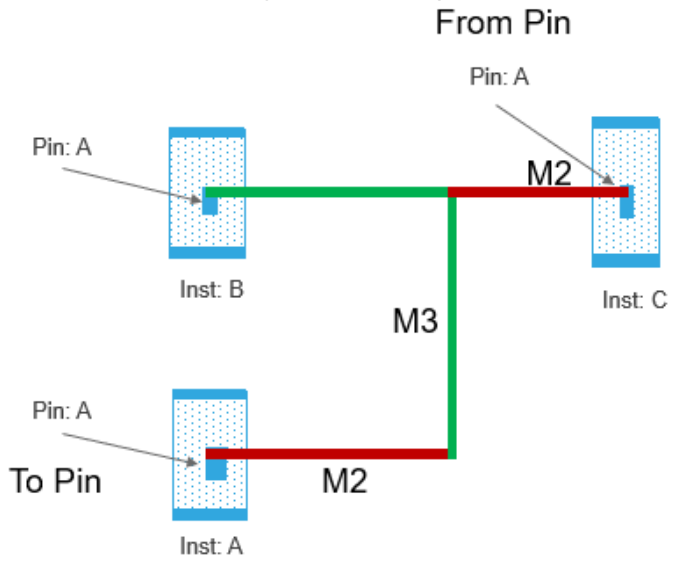
- The following command selects all wires on the *METAL1* and *METAL2* layers:
  ```
  select_routes -layer {METAL1 METAL2}
  ```
- The following command selects all vias on the cut layer `VIA12`:
  ```
  select_routes -layer VIA12 -obj_type via
  ```
- The following command selects all vias on the cut layers `VIA12` and `VIA23`:
  ```
  select_routes -layer {VIA12 VIA23} -obj_type via
  ```
- The following command selects all wires on the *METAL1* and *METAL2* layers and vias on the cut layer `VIA12`:
  ```
  select_routes -layer {METAL1 METAL2 VIA12} -obj_type {wire via}
  ```
- The following command selects all horizontal fillwires on the *METAL1* layer:
  ```
  select_routes -layer METAL1 -direction h -shapes fillwire
  ```
- The following command selects VDD STRIPEs with subclass "stripe_A":
  ```
  select_routes -nets VDD -shapes stripe -type special -sub_class stripe_A
  ```
- The following command selects all vias of shape `FOLLOWPIN` with names that match the pattern `tapcellvia_VIA1_RUL*`:
  ```
  select_routes -shapes followpin -via_cell tapcellvia_VIA1_RUL* -obj_type via
  ```
- The following command selects all the modified wires and vias:
  ```
  select_routes -with_edit_flag 1 -obj_type {wire via}
  ```
- The following command selects the complete wire between the specified from and to pins:
  ```
  select_routes -from_pin C/A -to_pin A/A
  ```
  The selected wire is depicted in red in the diagram below:



- The following command selects only the wire segments on the `M2` layer between the specified from and to pins:

  ```
  select_routes -from_pin C/A -to_pin A/A -layer M2
  ```

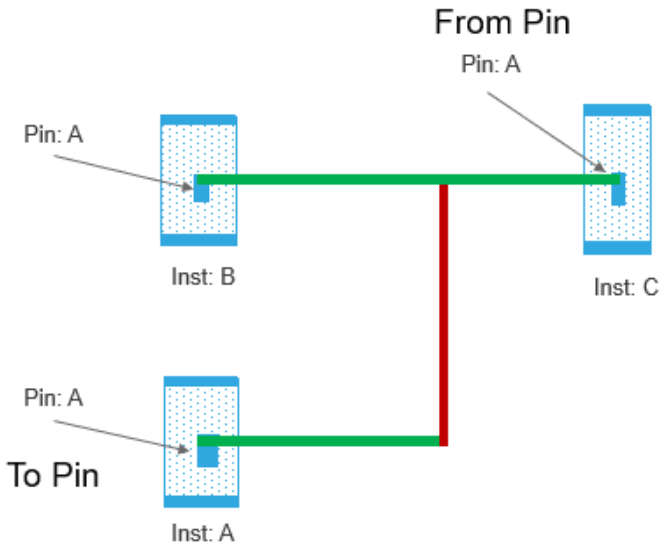
  The selected wire segments are depicted in red in the diagram below:



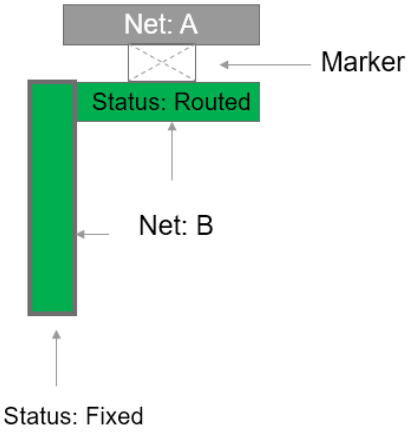
- The following command only the vertical wire segments between the specified from and to pins:

  ```
  select_routes -from_pin C/A -to_pin A/A -direction V
  ```

  The selected wire segment is depicted in red in the diagram below:

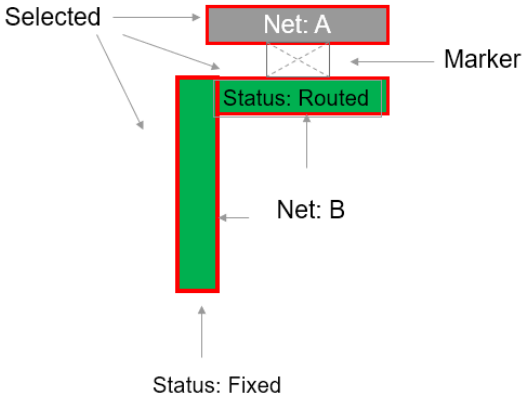


- Consider the example below:



  - The following command selects all nets that have DRC markers:

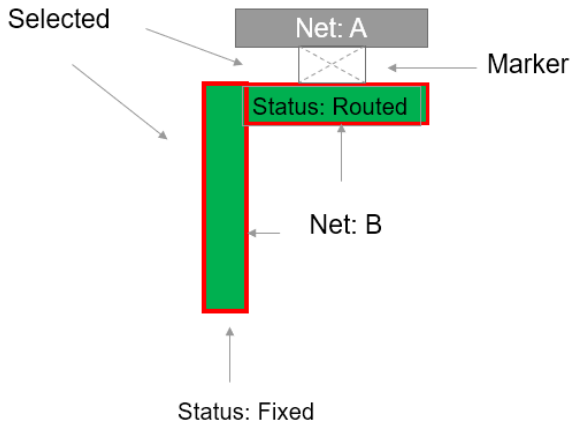
```
select_routes -regular_wire_with_drc
```



Both nets A and B are selected.

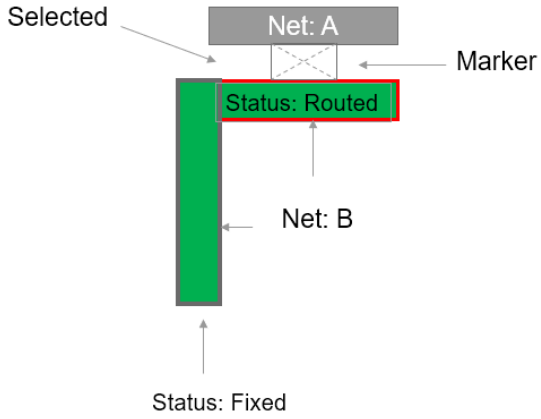
- The following command selects wires with DRC markers from only Net B:

  ```
  select_routes -nets B -regular_wire_with_drc
  ```



- The following command selects only routed wires with DRC markers from Net B:

  ```
  select_routes -nets B -regular_wire_with_drc -status ROUTED
  ```



- The following set of commands select and delete nets with DRC markers while excluding clock nets:

  ```
  select_routes -regular_wire_with_drc
  deselect_routes -use clock
  delete_routes -selected
  ```

Related Information

- `delete_routes`
- `deselect_routes`
- `edit_add_route_point`
- `edit_adjust_routes`
- `edit_cut_route`
- `edit_duplicate_routes`
- `edit_end_route_point`
- `edit_merge_routes`
- `edit_move_routes`
- `edit_resize_routes`
- `edit_rotate`
- `edit_split_routes`
- `edit_trim_routes`
- `edit_update_route_layer`
- `edit_update_route_net`
- `edit_update_route_rule`
- `edit_update_route_status`
- `edit_update_via`
- `gui_redo`
- `gui_undo`
- `set_db`
- Attributes in the `edit_wire` category
- Editing Wires chapter in the *User Guide*
- Edit Menu chapter in the *Menu Reference*
    - *Wire* section

select_secondary_pg_net

```
select_secondary_pg_net
[-help]
-insts inst+
{-net net | -pin pin_name}
```

Selects Power/Ground (P/G) net wire segments from specified instances. The wire segments are selected from the instance P/G pins to stripes (swire targets). This provides an easy way to select and delete routing from specific instance P/G pins to stripes.

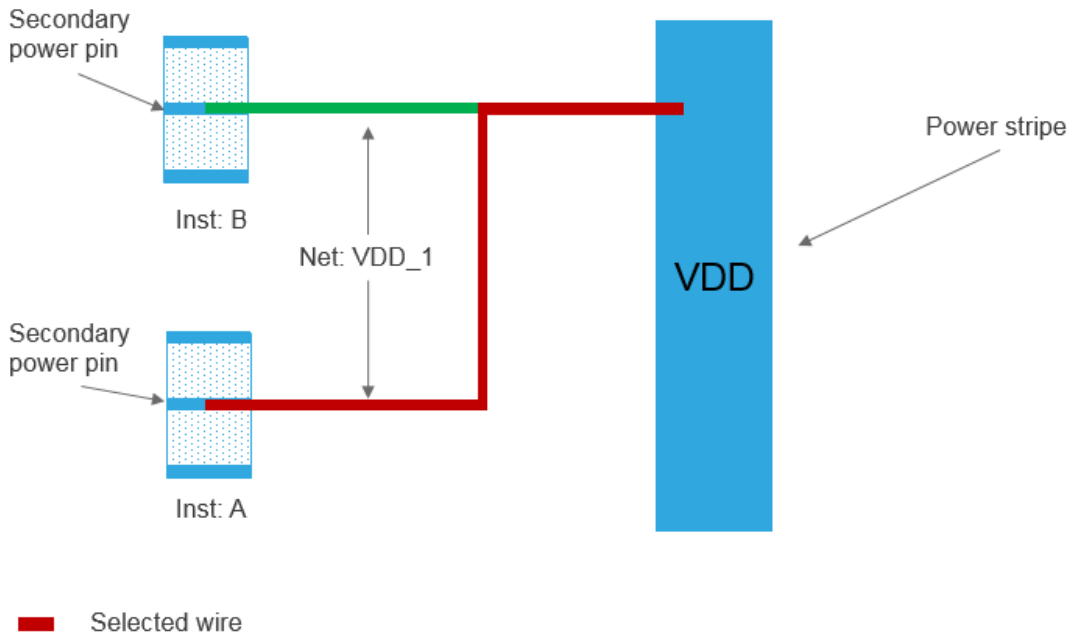
Note: This command does not support dangling secondary P/G nets.

Parameters

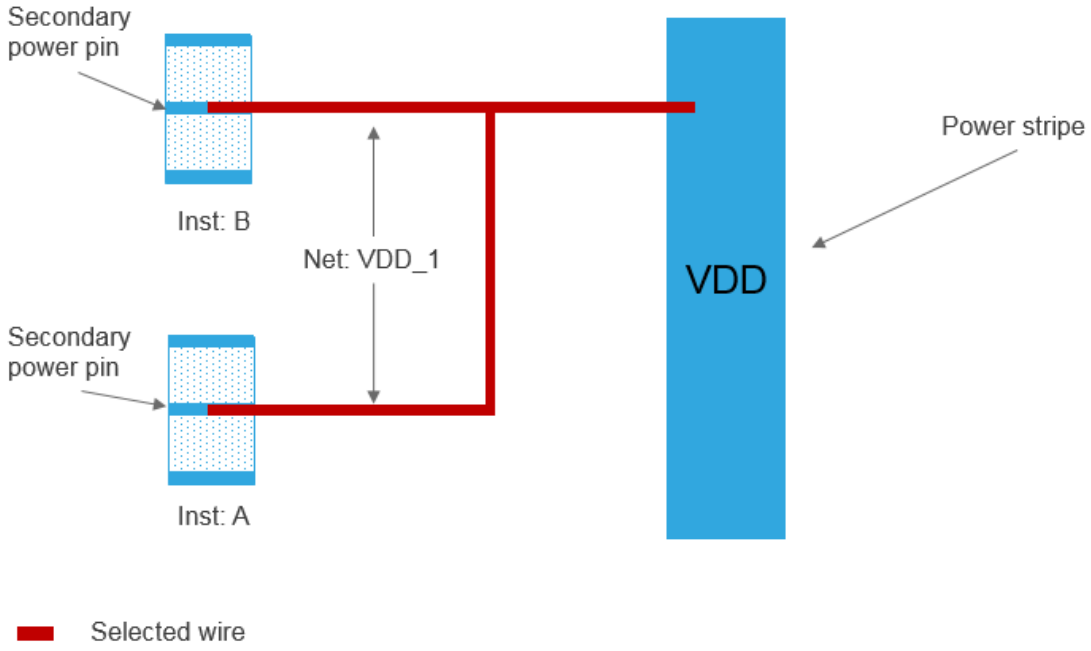
`-help`	Prints the usage of the command. For a detailed description of the command, use the `man` command: `man select_secondary_pg_net`
`-insts inst+`	Specifies the list of instances that have a secondary PG pin. Wildcards are supported in instance names. If one or more specified instances does not connect with the specified net, the tool displays a warning and ignores those instances. *Data_type*: (inst)+, required
`-net net_name`	Specifies the net name. **Note**: *net_name* does not take any wildcard values. This parameter cannot be used with `-pin`. *Data_type*: (net), optional
`-pin pin_name`	Specifies the pin name. This parameter cannot be used with `-net`. *Data_type*: string, optional

Examples

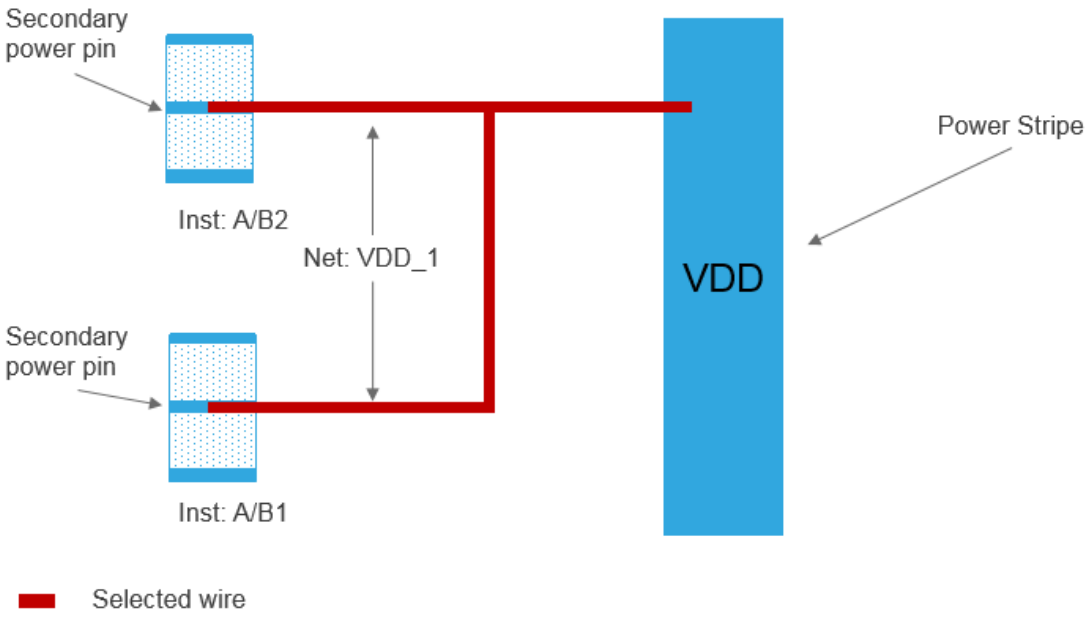
- The following command selects the routing from the P/G pin in instance `A` of net `VDD_1` to the power stripe:
  ```
  select_secondary_pg_net -net VDD_1 -insts {A}
  ```



- The following command selects the routing from the P/G pins in instances `A` and `B` in net `VDD_1` to the power stripe:
```
select_secondary_pg_net -net VDD_1 -insts {A B}
```



- The following command selects the routing from the P/G pins in instances `A/B1` and A/B2 in net `VDD_1` to the power stripe:
```
select_secondary_pg_net -net VDD_1 -insts {A/*}
```



- The following command selects secondary PG net through the `VDDO` pin and checks all its ports:
  ```
  select_secondary_pg_net -insts $instn -pin VDDO
  ```
  If a port of the the `VDDO` pin has an open, the tool prints a message as follows:
  ```
  **WARN: (IMPSPR-405): There is no route for secondary pin "VDDO" to
  inst "PD_CHIPLET2_PADS_PD_CHIPPADS_pads_out_balanced_ls_padring_chiplet2_u_DISP_LSYNC_C_UPF_LS"
  at '3.085500 1.902500'.
  ```

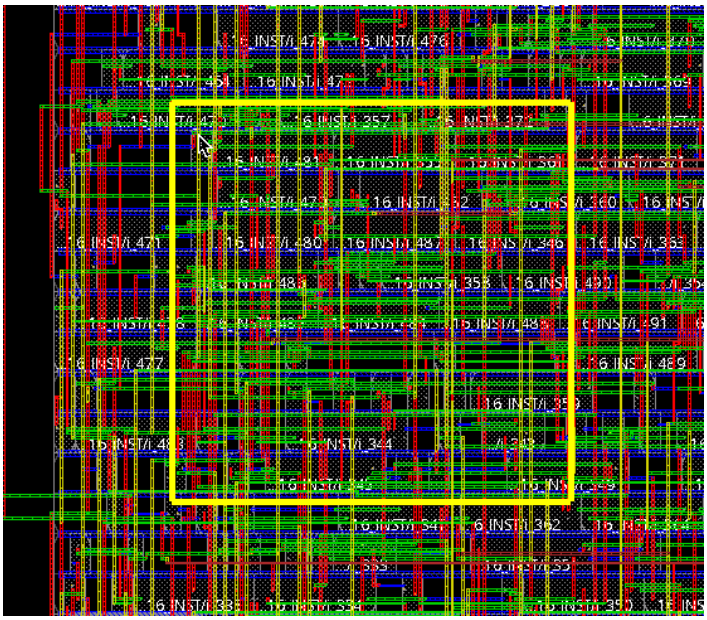
Related Information

- `delete_secondary_pg_net`
- `deselect_secondary_pg_net`
- Editing Wires chapter in the *User Guide*
- Edit Menu chapter in the *Menu Reference*
  - *Wire* section

start_parallel_edit

```
start_parallel_edit
[-help]
[-area_restricted]
-region {x1 y1 x2 y2}
```

Initializes parallel editing by drawing a yellow square on the main window to indicate the edit area and saving the physical data, net attributes, via cell names and Non-Default Rules (NDRs) to multiple binary files.



Parameters

`-help`	Prints a brief description that includes type and default information for each `start_parallel_edit` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man start_parallel_edit`
`-area_restricted`	Writes out only the different objects inside or touching the specified edit area. This is a strict interpretation of the region, and ignores the changes made outside of the region. *Data_type*: bool, optional
`-region {x1 y1 x2 y2}`	Specifies the coordinates of the edit area. *Data_type*: rect, required

Example

- The following command marks the specified edit area in yellow and saves the design data to binary files:

```
innovus 8> start_parallel_edit -area_restricted -region {12.524 6.627 0.0 0.0}
Saving net attribute and regular wire/via information
Saving instance information
Saving physical pin information
Saving NDR rule information
Saving routing blockage information
Saving placement blockage information
Saving special wire/via information
```

Related Information

- `end_parallel_edit`
- `read_parallel_edit_files`
- Editing Wires chapter in the *User Guide*
  - *Parallel Editing Capability*

47

Metric Commands and Attributes

- Common Metric Names
- create_metric_page
- create_metric_table
- create_metric_table_heading
- create_snapshot
- define_metric
- delete_metric
- delete_metric_page
- delete_metric_run
- delete_metric_table
- delete_metric_table_heading
- get_metric
- get_metric_alias
- get_metric_config
- get_metric_definition
- get_metric_header
- metric Category Attributes
- pop_snapshot_stack
- push_snapshot_stack
- read_metric
- read_metric_config
- read_metric_html_package
- report_metric
- report_qor
- reset_metric_config
- reset_metric_run_id
- run_metric_category
- set_metric

- set_metric_alias
- set_metric_header
- update_metric
- update_metric_definition
- write_metric
- write_metric_config
- rename_snapshot
- delete_metric_header

Common Metric Names

Clock Metrics

The following table provides the list of various clock metrics.

Metric Object Name	Description
`clock.area.buffer` `units: um^2`	Reports the area of clock buffer instances. Data Precision: 2 A smaller number is better when doing HTML comparison.
`clock.area.buffer.<clock>` `units: um^2`	Reports the area of clock buffer instances per clock. Data Precision: 2 A smaller number is better when doing HTML comparison.
`clock.area.clkgate` `units: um^2`	Reports the area of ICG instances. Data Precision: 2 A smaller number is better when doing HTML comparison.
`clock.area.clkgate.<clock>` `units: um^2`	Reports the area of ICG instances per clock. Data Precision: 2 A smaller number is better when doing HTML comparison.
`clock.area.inverter` `units: um^2`	Reports the area of clock inverter instances. Data Precision: 2 A smaller number is better when doing HTML comparison.
`clock.area.inverter.<clock>` `units: um^2`	Reports the area of clock inverter instances per clock. Data Precision: 2 A smaller number is better when doing HTML comparison.

`clock.area.total` `units: um^2`	Reports the total clock area. Data Precision: 2 A smaller number is better when doing HTML comparison.
`clock.area.total.<clock>` `units: um^2`	Reports the total clock area per clock. Data Precision: 2 A smaller number is better when doing HTML comparison.
`clock.capacitance.total`	Reports the total clock (net+pin) capacitance. A smaller number is better when doing HTML comparison.
`clock.capacitance.total.<clock>`	Reports the total clock (net+pin) capacitance per clock. A smaller number is better when doing HTML comparison.
`clock.drv.max_cap`	Reports the total clock max capacitance violations for primary delay corner. A smaller number is better when doing HTML comparison.
`clock.drv.max_cap.total.<clock>.<delay_corner>.<early|late>`	Reports the total clock max capacitance violations per clock. A smaller number is better when doing HTML comparison.
`clock.drv.max_cap.worst`	Reports the worst clock max capacitance violation for primary delay corner. A smaller number is better when doing HTML comparison.
`clock.drv.max_cap.worst.<clock>.<delay_corner>.<early|late>`	Reports the worst clock max capacitance violation per clock. A smaller number is better when doing HTML comparison.
`clock.drv.max_tran.total.<type>.<clock>.<delay_corner>.<early|late>`	Reports the total clock max transition violations per clock per delay corner. A smaller number is better when doing HTML comparison.
`clock.drv.max_tran.worst`	Reports the worst clock max transition violation for primary delay corner. A smaller number is better when doing HTML comparison.
`clock.drv.max_tran.worst.<clock>.<delay_corner>.<early|late>`	Reports the worst clock max transition violation per clock per delay corner. A smaller number is better when doing HTML comparison.
`clock.flexible_htree:<flexible_htree>.instances`	Reports the number instances of the flexible H-tree (excluding sinks). Data Precision: 0

`clock.flexible_htree:<flexible_htree>.latency.<max>`	Reports the latency in the primary half-corner of the flexible H-tree. Data Precision: 3 A smaller number above target is better when doing HTML comparison.
`clock.flexible_htree:<flexible_htree>.levels`	Reports the levels of the flexible H-tree (excluding H-tree sinks). Data Precision: 0
`clock.flexible_htree:<flexible_htree>.preferred_layer_adherence`	Reports the preferred layer adherence ratio of nets of the flexible H-tree. Data Precision: 3
`clock.flexible_htree:<flexible_htree>.sinks`	Reports the number of sinks of the flexible H-tree. Data Precision: 0
`clock.flexible_htree:<flexible_htree>.skew`	Reports the skew in the primary delay half-corner of the flexible H-tree's skew group. Data Precision: 3 A smaller number above target is better when doing HTML comparison.
`clock.flexible_htree:<flexible_htree>.vias`	Reports the number of vias of nets of the flexible H-tree. Data Precision: 0
`clock.flexible_htree:<flexible_htree>.wire_length`	Reports the total wire length of nets of the flexible H-tree. Data Precision: 0
`clock.instances.buffer`	Reports the number of clock buffer instances. A smaller number is better when doing HTML comparison.
`clock.instances.buffer.<clock>`	Reports the number of clock buffer instances per clock. A smaller number is better when doing HTML comparison.
`clock.instances.buffer_and_inverter`	Reports the total number of total clock buffers and inverters. Data Precision: 0 A smaller number is better when doing HTML comparison. internal 1
`clock.instances.clkgate`	Reports the number of ICG instances. A smaller number is better when doing HTML comparison.
`clock.instances.clkgate.<clock>`	Reports the number of ICG instances per clock. A smaller number is better when doing HTML comparison.

`clock.instances.inverter`	Reports the number of clock inverter instances. A smaller number is better when doing HTML comparison.
`clock.instances.inverter.<clock>`	Reports the number of clock inverter instances per clock. A smaller number is better when doing HTML comparison.
`clock.instances.total`	Reports the total number of clock instances. A smaller number is better when doing HTML comparison.
`clock.instances.total.<clock>`	Reports the total number of clock instances per clock. A smaller number is better when doing HTML comparison.
`clock.nets.length.total units: um`	Reports the total length of clock nets. Data Precision: 0 A smaller number is better when doing HTML comparison.
`clock.skew.max`	Reports the maximum global clock skew. A smaller number is better when doing HTML comparison.
`clock.skew.max.<clock>`	Reports the maximum global clock skew per clock.
`clock.skew_groups`	Reports the list of skew groups.
`clock.timing.max_latency.<clock>`	Reports the maximum network latency. A smaller number is better when doing HTML comparison.
`clock.timing.min_latency.<clock>`	Reports the minimum network latency. A smaller number is better when doing HTML comparison.
`clock.timing.period.<clock>`	Reports the clock period. A smaller number is better when doing HTML comparison.
`clock.timing.source_latency.<clock>`	Reports the source latency.
`clock.timing.uncertainty.<clock>`	Reports the clock uncertainty.

Design Metrics

The following table provides the list of various design metrics.

Metric Object Name	Description

`design.area`
`units: um^2`
`type area`
	Total design area
	Data Precision: 0
	A smaller number is better when doing HTML comparisons.

design.area.always_on units: um^2 type area	Total area of always on instances Data Precision: 0 A smaller number is better when doing HTML comparisons.
design.area.area_io units: um^2 type area	Total area of area_io instances Data Precision: 0 A smaller number is better when doing HTML comparisons.
design.area.blackbox units: um^2 type area	Total area of block instances Data Precision: 0 A smaller number is better when doing HTML comparisons.
design.area.block units: um^2 type area	Total area of block instances Data Precision: 0 A smaller number is better when doing HTML comparisons.
design.area.block.<block> units: um^2 type area	Design area per block Data Precision: 0 A smaller number is better when doing HTML comparisons.
design.area.buffer units: um^2 type area	Total area of buffer instances Data Precision: 0 A smaller number is better when doing HTML comparisons.
design.area.combinatorial units: um^2 type area	Total area of combinational standard cell logic Data Precision: 0 A smaller number is better when doing HTML comparisons.
design.area.filler units: um^2 type area	Total design area of filler intances Data Precision: 0 A smaller number is better when doing HTML comparisons.
design.area.hinst:<block> units: um^2 type area	Design area per block Data Precision: 0 A smaller number is better when doing HTML comparisons.
design.area.icg units: um^2 type area	Total area of icg instances Data Precision: 0 A smaller number is better when doing HTML comparisons.

design.area.inverter units: um^2 type area	Total area of inverter Data Precision: 0 A smaller number is better when doing HTML comparisons.
design.area.io units: um^2 type area	Total area of io instances Data Precision: 0 A smaller number is better when doing HTML comparisons.
design.area.iso_ls units: um^2 type area	Total area of iso and level shifter instances Data Precision: 0 A smaller number is better when doing HTML comparisons.
design.area.latch units: um^2 type area	Total area of latch instances Data Precision: 0 A smaller number is better when doing HTML comparisons.
design.area.logical units: um^2 type area	Total area of logical Data Precision: 0 A smaller number is better when doing HTML comparisons.
design.area.macro units: um^2 type area	Total area of block instances Data Precision: 0 A smaller number is better when doing HTML comparisons.
design.area.physical units: um^2 type area	Total area of physical only instances Data Precision: 0 A smaller number is better when doing HTML comparisons.
design.area.power_switch units: um^2 type area	Total area of power switches instances Data Precision: 0 A smaller number is better when doing HTML comparisons.
design.area.register units: um^2 type area	Total area of register instances Data Precision: 0 A smaller number is better when doing HTML comparisons.
design.area.sequential units: um^2 type area	Total area of sequential instances Data Precision: 0 A smaller number is better when doing HTML comparisons.

`design.area.std_cell`
`units: um^2`
`type area`
	Total area of standard cell instances
	Data Precision: 0
	A smaller number is better when doing HTML comparisons.

`design.bin.density.peak` `units: %`	Peak bin density based on PSP report Data Precision: 1 A smaller number is better when doing HTML comparisons.internal 1
`design.bin.density.top.010` `units: %`	The average bin density of top 10 bins Data Precision: 1 A smaller number is better when doing HTML comparisons.internal 1
`design.bin.density.top.0100` `units: %`	The average bin density of top 100 bins Data Precision: 1 A smaller number is better when doing HTML comparisons.internal 1
`design.bin.density.top.1` `units: %`	The average bin density of top 1 percent bins among all bins Data Precision: 1 A smaller number is better when doing HTML comparisons.internal 1
`design.bin.density.top.10` `units: %`	The average bin density of top 10 percent bins among all bins Data Precision: 1 A smaller number is better when doing HTML comparisons.internal 1
`design.bin.density.top.2` `units: %`	The average bin density of top 2 percent bins among all bins Data Precision: 1 A smaller number is better when doing HTML comparisons.internal 1
`design.bin.density.top.20` `units: %`	The average bin density of top 20 percent bins among all bins Data Precision: 1 A smaller number is better when doing HTML comparisons.internal 1
`design.bin.density.top.5` `units: %`	The average bin density of top 5 percent bins among all bins Data Precision: 1 A smaller number is better when doing HTML comparisons.internal 1
`design.bin.density.viol.bin`	The number of bins that violate the bin density constraints based on PSP report Data Precision: 0 A smaller number is better when doing HTML comparisons.internal 1
`design.bin.density.viol.localrow`	The number of local rows that violate the bin density constraints based on PSP report Data Precision: 0 A smaller number is better when doing HTML comparisons.internal 1

`design.bin.density.viol.sbin`	The number of sbins that violate the bin density constraints based on PSP report Data Precision: 0 A smaller number is better when doing HTML comparisons.internal 1
`design.bin.density.viol.subrow`	The number of subrows that violate the bin density constraints based on PSP report Data Precision: 0 A smaller number is better when doing HTML comparisons.internal 1
`design.bin.viol.100`	The number of bins that have densities more than 100 percent Data Precision: 0 A smaller number is better when doing HTML comparisons.internal 1
`design.bin.viol.60`	The number of bins that have densities more than 60 percent Data Precision: 0 A smaller number is better when doing HTML comparisons.internal 1
`design.bin.viol.70`	The number of bins that have densities more than 70 percent Data Precision: 0 A smaller number is better when doing HTML comparisons.internal 1
`design.bin.viol.80`	The number of bins that have densities more than 80 percent Data Precision: 0 A smaller number is better when doing HTML comparisons.internal 1
`design.bin.viol.90`	The number of bins that have densities more than 90 percent Data Precision: 0 A smaller number is better when doing HTML comparisons.internal 1
`design.blockages.place.area`	Placement blockage area Data Precision: 0
`design.blockages.route.area`	Routing blockage area Data Precision: 0
`design.blockages.route.area.layer:<layer>`	Routing blockage area per layer Data Precision: 0
`design.cell.viol.boundary`	The number of cells that are outside of design boundary Data Precision: 0 A smaller number is better when doing HTML comparisons.internal 1
`design.cell.viol.macro`	The number of cells that are completely blocked by macros Data Precision: 0 A smaller number is better when doing HTML comparisons.internal 1

`design.congestion.hotspot.max`	Maximum congestion hotspot size Data Precision: 2 A smaller number is better when doing HTML comparisons.
`design.congestion.hotspot.total`	Total size of all congestion hotspots Data Precision: 2 A smaller number is better when doing HTML comparisons.

`design.density`
`units: %`
`type density`
	Density w/o filler cells
	Data Precision: 1 A smaller number is better when doing HTML comparisons.

`design.density.peak`
`units: %`
`type density`
	Peak design density w/o filler cells
	Data Precision: 1 A smaller number is better when doing HTML comparisons.

`design.floorplan.image` `type imageplot`	Design floorplan image Data Precision: 0
`design.instance.always_on tcl`	Total number of always on instances auto_calculated in um::set_additional_metrics A smaller number is better when doing HTML comparisons.
`design.instance.power_switch`	Total number of power switches A smaller number is better when doing HTML comparisons.
`design.instances`	Total number of instances Data Precision: 0 A smaller number is better when doing HTML comparisons.
`design.instances.always_on`	Total number of always on instances Data Precision: 0 A smaller number is better when doing HTML comparisons.
`design.instances.area_io`	Total number of Area IO instances Data Precision: 0 A smaller number is better when doing HTML comparisons.
`design.instances.blackbox`	Total number of blackbox instances Data Precision: 0 A smaller number is better when doing HTML comparisons.

`design.instances.block`	Total number of block instances Data Precision: 0 A smaller number is better when doing HTML comparisons.
`design.instances.block.<block>`	Number of instances per block Data Precision: 0 A smaller number is better when doing HTML comparisons.
`design.instances.buffer`	Total number of buffer instances Data Precision: 0 A smaller number is better when doing HTML comparisons.
`design.instances.combinatorial`	Total number of combinational standard cell instances Data Precision: 0 A smaller number is better when doing HTML comparisons.
`design.instances.filler`	Total number of filler instances Data Precision: 0 A smaller number is better when doing HTML comparisons.
`design.instances.hinst:<block>`	Number of instances per block Data Precision: 0 A smaller number is better when doing HTML comparisons.
`design.instances.icg`	Total number of clock gating instances Data Precision: 0 A smaller number is better when doing HTML comparisons.
`design.instances.inverter`	Total number of inverter instances Data Precision: 0 A smaller number is better when doing HTML comparisons.
`design.instances.io`	Total number of IO instances Data Precision: 0 A smaller number is better when doing HTML comparisons.
`design.instances.iso_ls`	Total area of iso and level shifter instances Data Precision: 0 A smaller number is better when doing HTML comparisons.

`design.instances.latch`	Total number of latch instances Data Precision: 0 A smaller number is better when doing HTML comparisons.
`design.instances.logical`	Total number of logical instances Data Precision: 0 A smaller number is better when doing HTML comparisons.
`design.instances.macro`	Total number of block instances Data Precision: 0 A smaller number is better when doing HTML comparisons.
`design.instances.physical`	Total number of physical instances Data Precision: 0 A smaller number is better when doing HTML comparisons.
`design.instances.power_switch`	Total number of power switches Data Precision: 0 A smaller number is better when doing HTML comparisons.
`design.instances.register`	Total number of sequential instances Data Precision: 0 A smaller number is better when doing HTML comparisons.
`design.instances.sequential`	Total number of sequential instances Data Precision: 0 A smaller number is better when doing HTML comparisons.
`design.instances.std_cell`	Total number of standard cell instances Data Precision: 0 A smaller number is better when doing HTML comparisons.
`design.movement.avg` `units: site`	Average cell movement Data Precision: 0 A smaller number is better when doing HTML comparisons.internal 1
`design.movement.cell`	Total number of moved cells Data Precision: 0 A smaller number is better when doing HTML comparisons.internal 1

`design.movement.clkbuf.avg` `units: site`	Average movement Data Precision: 0 A smaller number is better when doing HTML comparisons.internal 1
`design.movement.clkbuf.max` `units: site`	Max movement Data Precision: 0 A smaller number is better when doing HTML comparisons. internal 1
`design.movement.clkbuf.moved`	Total number of moved clock buffers and inverters Data Precision: 0 A smaller number is better when doing HTML comparisons.internal 1
`design.movement.clkgate.avg` `units: site`	Average movement Data Precision: 0 A smaller number is better when doing HTML comparisons.internal 1
`design.movement.clkgate.max` `units: site`	Max movement Data Precision: 0 A smaller number is better when doing HTML comparisons.internal 1
`design.movement.clkgate.moved`	Total number of moved ICG instances Data Precision: 0 A smaller number is better when doing HTML comparisons.internal 1
`design.movement.max` `units: site`	Max cell movement Data Precision: 0 A smaller number is better when doing HTML comparisons.internal 1
`design.movement.movable`	Number of movable cells Data Precision: 0 A smaller number is better when doing HTML comparisons.internal 1
`design.movement.moved`	Total number of moved cells Data Precision: 0 A smaller number is better when doing HTML comparisons.internal 1
`design.movement.sequential.avg` `units: site`	Average movement Data Precision: 0 A smaller number is better when doing HTML comparisons.internal 1

`design.movement.sequential.max` `units: site`	Max movement Data Precision: 0 A smaller number is better when doing HTML comparisons.internal 1
`design.movement.sequential.moved`	Total number of moved sequential instances Data Precision: 0 A smaller number is better when doing HTML comparisons.internal 1
`design.movement.top.0.05` `units: site`	Top 0.05 percent movement among all cell movements Data Precision: 0 A smaller number is better when doing HTML comparisons.internal 1
`design.movement.top.0.1` `units: site`	Top 0.1 percent movement among all cell movements Data Precision: 0 A smaller number is better when doing HTML comparisons.internal 1
`design.movement.top.0.2` `units: site`	Top 0.2 percent movement among all cell movements Data Precision: 0 A smaller number is better when doing HTML comparisons.internal 1
`design.movement.top.0.5` `units: site`	Top 0.5 percent movement among all cell movements Data Precision: 0 A smaller number is better when doing HTML comparisons.internal 1
`design.movement.top.1` `units: site`	Top 1 percent movement among all cell movements Data Precision: 0 A smaller number is better when doing HTML comparisons.internal 1
`design.movement.viol`	The number of cells that violates the max movement constraint Data Precision: 0 A smaller number is better when doing HTML comparisons.internal 1
`design.multibit.avg_width`	Total bits / total flops Data Precision: 0 A larger number is better when doing HTML comparisons.
`design.multibit.count`	Total number of multi-bit instances Data Precision: 0 A larger number is better when doing HTML comparisons.

`design.multibit.coverage` `units: %`	Total multi-bits / total bits Data Precision: 1 A larger number is better when doing HTML comparisons.
`design.name` `type area`	Design name Data Precision: 0
`design.nets.avg_pin`	Average terms per net Data Precision: 1 A smaller number is better when doing HTML comparisons.
`design.nets.global`	Number of global nets Data Precision: 0 A smaller number is better when doing HTML comparisons.
`design.nets.missing_iso`	Number of nets crossing domains w/o ISO Data Precision: 0 A smaller number is better when doing HTML comparisons.
`design.nets.signal`	Number of signal nets Data Precision: 0 A smaller number is better when doing HTML comparisons.
`design.overlapping.area.all`	The number of sites that have more than two cells placed on it based on PSP report Data Precision: 0 A smaller number is better when doing HTML comparisons.internal 1
`design.overlapping.area.nopad`	The number of sites that have more than two non-padded cell area placed on it Data Precision: 0 A smaller number is better when doing HTML comparisons.internal 1
`design.overlapping.cost`	The cost of sites that have overlapping cells based on PSP report Data Precision: 0 A smaller number is better when doing HTML comparisons.internal 1
`design.pairpin.ratio.top.1` `units: %`	The average pairpin ratio inside a bin of top 1 percent bins among all bins Data Precision: 1 A smaller number is better when doing HTML comparisons.internal 1
`design.pairpin.ratio.top.10` `units: %`	The average pairpin ratio inside a bin of top 10 percent bins among all bins Data Precision: 1 A smaller number is better when doing HTML comparisons.internal 1

design.pairpin.ratio.top.2 units: %	The average pairpin ratio inside a bin of top 2 percent bins among all bins Data Precision: 1 A smaller number is better when doing HTML comparisons.internal 1
design.pairpin.ratio.top.20 units: %	The average pairpin ratio inside a bin of top 20 percent bins among all bins Data Precision: 1 A smaller number is better when doing HTML comparisons.internal 1
design.pairpin.ratio.top.5 units: %	The average pairpin ratio inside a bin of top 5 percent bins among all bins Data Precision: 1 A smaller number is better when doing HTML comparisons.internal 1
design.pairspacing.percentage.0 units: %	The percentage of cell pairs with spacing 0 Data Precision: 1 internal 1
design.pairspacing.percentage.1 units: %	The percentage of cell pairs with spacing 1 Data Precision: 1 internal 1
design.pairspacing.percentage.2 units: %	The percentage of cell pairs with spacing 2 Data Precision: 1 internal 1
design.pairspacing.percentage.3 units: %	The percentage of cell pairs with spacing 3 Data Precision: 1 internal 1
design.pairspacing.percentage.4 units: %	The percentage of cell pairs with spacing 4 Data Precision: 1 internal 1
design.pairspacing.percentage.5 units: %	The percentage of cell pairs with spacing 5 Data Precision: 1 internal 1
design.pairspacing.percentage.6 units: %	The percentage of cell pairs with spacing 6 Data Precision: 1 internal 1
design.pairspacing.percentage.7 units: %	The percentage of cell pairs with spacing 7 Data Precision: 1 internal 1
design.pin.density.peak units: %	Peak pin density inside a bin based on PSP report Data Precision: 1 A smaller number is better when doing HTML comparisons.internal 1
design.pin.density.top.1 units: %	The average pin density inside a bin of top 1 percent bins among all bins Data Precision: 1 A smaller number is better when doing HTML comparisons.internal 1

`design.pin.density.top.10` `units: %`	The average pin density inside a bin of top 10 percent bins among all bins Data Precision: 1 A smaller number is better when doing HTML comparisons.internal 1
`design.pin.density.top.2` `units: %`	The average pin density inside a bin of top 2 percent bins among all bins Data Precision: 1 A smaller number is better when doing HTML comparisons.internal 1
`design.pin.density.top.20` `units: %`	The average pin density inside a bin of top 20 percent bins among all bins Data Precision: 1 A smaller number is better when doing HTML comparisons.internal 1
`design.pin.density.top.5` `units: %`	The average pin density inside a bin of top 5 percent bins among all bins Data Precision: 1 A smaller number is better when doing HTML comparisons.internal 1
`design.place.drc.image` `type imageplot`	Design place DRC image. Data Precision: 0
`design.power_domains`	List of power domains Data Precision: 0
`design.psp.hpwl`	HPWL wirelength based on PSP report Data Precision: 0 A smaller number is better when doing HTML comparisons.internal 1
`design.psp.hpwl.crit`	HPWL wirelength for critical nets based on PSP report Data Precision: 0 A smaller number is better when doing HTML comparisons.internal 1
`design.psp.hpwl.crit.w`	Weighted HPWL wirelength for critical nets based on PSP report Data Precision: 0 A smaller number is better when doing HTML comparisons.internal 1
`design.psp.hpwl.w`	Weighted HPWL wirelength based on PSP report Data Precision: 0 A smaller number is better when doing HTML comparisons.internal 1
`design.psp.mst`	MST wirelength based on PSP report Data Precision: 0 A smaller number is better when doing HTML comparisons. internal 1

`design.psp.mst.crit`	MST wirelength for critical nets based on PSP report Data Precision: 0 A smaller number is better when doing HTML comparisons. internal 1
`design.psp.mst.crit.w`	Weighted MST wirelength for critical nets based on PSP report Data Precision: 0 A smaller number is better when doing HTML comparisons.internal 1
`design.psp.mst.w`	Weighted MST wirelength based on PSP report Data Precision: 0 A smaller number is better when doing HTML comparisons.internal 1
`design.route.drc.image` `type imageplot`	Design route image. Data Precision: 0
`design.sbin.density.peak` `units: %`	Peak sbin density based on PSP report Data Precision: 1 A smaller number is better when doing HTML comparisons.internal 1
`design.sbin.density.top.010` `units: %`	The average sbin density of top 10 sbins Data Precision: 1 A smaller number is better when doing HTML comparisons.internal 1
`design.sbin.density.top.0100` `units: %`	The average sbin density of top 100 sbins Data Precision: 1 A smaller number is better when doing HTML comparisons.internal 1
`design.sbin.density.top.1` `units: %`	The average sbin density of top 1 percent sbins among all sbins Data Precision: 1 A smaller number is better when doing HTML comparisons.internal 1
`design.sbin.density.top.10` `units: %`	The average sbin density of top 10 percent sbins among all sbins Data Precision: 1 A smaller number is better when doing HTML comparisons.internal 1
`design.sbin.density.top.2` `units: %`	The average sbin density of top 2 percent sbins among all sbins Data Precision: 1 A smaller number is better when doing HTML comparisons.internal 1
`design.sbin.density.top.20` `units: %`	The average sbin density of top 20 percent sbins among all sbins Data Precision: 1 A smaller number is better when doing HTML comparisons.internal 1

`design.sbin.density.top.5` `units: %`	The average sbin density of top 5 percent sbins among all sbins Data Precision: 1 A smaller number is better when doing HTML comparisons.internal 1
`design.sbin.viol.100`	The number of sbins that have densities more than 100 percent Data Precision: 0 A smaller number is better when doing HTML comparisons.internal 1
`design.sbin.viol.60`	The number of sbins that have densities more than 60 percent Data Precision: 0 A smaller number is better when doing HTML comparisons.internal 1
`design.sbin.viol.70`	The number of sbins that have densities more than 70 percent Data Precision: 0 A smaller number is better when doing HTML comparisons.internal 1
`design.sbin.viol.80`	The number of sbins that have densities more than 80 percent Data Precision: 0 A smaller number is better when doing HTML comparisons.internal 1
`design.sbin.viol.90`	The number of sbins that have densities more than 90 percent Data Precision: 0 A smaller number is better when doing HTML comparisons.internal 1
`design.sitearea.actual`	Actual site area Data Precision: 0 internal 1
`design.sitearea.padding`	Padded site area Data Precision: 0 internal 1
`design.sp.hpwl`	HPWL wirelength based on sp placement WL report Data Precision: 0 A smaller number is better when doing HTML comparisons.internal 1
`design.sp.steiner`	Formula based Steiner wirelength based on sp placement WL report Data Precision: 0 A smaller number is better when doing HTML comparisons.internal 1
`design.spin.density.peak` `units: %`	Peak pin density inside a sbin based on PSP report Data Precision: 1 A smaller number is better when doing HTML comparisons.internal 1

`design.spin.density.top.1` `units: %`	The average pin density inside a sbin of top 1 percent sbins among all sbins Data Precision: 1 A smaller number is better when doing HTML comparisons.internal 1
`design.spin.density.top.10` `units: %`	The average pin density inside a sbin of top 10 percent sbins among all sbins Data Precision: 1 A smaller number is better when doing HTML comparisons.internal 1
`design.spin.density.top.2` `units: %`	The average pin density inside a sbin of top 2 percent sbins among all sbins Data Precision: 1 A smaller number is better when doing HTML comparisons.internal 1
`design.spin.density.top.20` `units: %`	The average pin density inside a sbin of top 20 percent sbins among all sbins Data Precision: 1 A smaller number is better when doing HTML comparisons.internal 1
`design.spin.density.top.5` `units: %`	The average pin density inside a sbin of top 5 percent sbins among all sbins Data Precision: 1 A smaller number is better when doing HTML comparisons.internal 1

Flow Metrics

The following table provides the list of various flow metrics.

Metric Object Name	Description

`flow.cputime`
`units: s`
`type time`
	Returns the CPU time since the last snapshot
	Data Precision: 0
	A smaller number is better when doing HTML comparisons.

`flow.cputime.not_child`
`units: s`
`type time`
	Returns the CPU time taken since the last snapshot excluding child snapshots
	Data Precision: 0
	A smaller number is better when doing HTML comparisons.

`flow.cputime.total`
`units: s`
`type time`
	Returns the total CPU time
	Data Precision: 0
	A smaller number is better when doing HTML comparisons.

`flow.cputime.user`
`units: s`
`type time`
	Returns the user CPU time since the last snapshot Data Precision: 0 A smaller number is better when doing HTML comparisons.

`flow.cputime.user.not_child`
`units: s`
`type time`
	Returns the user CPU time since the last snapshot excluding child snapshots Data Precision: 0 A smaller number is better when doing HTML comparisons.

`flow.cputime.user.total`
`units: s`
`type time`
	Returns the total user CPU time
	Data Precision: 0
	A smaller number is better when doing HTML comparisons.

`flow.exclude_time_metric` `type bool`	Returns the CPU and Wall time are excluded from parent snapshot Data Precision: 0 inheritance_enabled 0
`flow.machine tcl`	Returns the info hostname description Hostname Data Precision: 0
`flow.machine.load`	Returns the host load average. Data Precision: 2 A smaller number is better when doing HTML comparisons.

`flow.memory`
`units: MB`
`type memory`
	Returns the current memory footprint
	Data Precision: 0
	A smaller number is better when doing HTML comparisons.

`flow.memory.instant`
`units: MB`
`type memory`
	Returns the current memory footprint
	Data Precision: 0
	A smaller number is better when doing HTML comparisons. inheritance_enabled 0

`flow.memory.resident`
`units: MB`
`type memory`
	Returns the current resident memory footprint
	Data Precision: 0
	A smaller number is better when doing HTML comparisons. inheritance_tcl

`flow.memory.resident.instant`
`units: MB`
`type memory`
	Returns the current resident memory footprint
	Data Precision: 0
	A smaller number is better when doing HTML comparisons. inheritance_enabled 0

`flow.memory.resident.peak`
`units: MB`
`type memory`
	Returns the maximum resident memory footprint
	Data Precision: 0
	A smaller number is better when doing HTML comparisons. inheritance_tcl

`flow.memory.resident.peak.instant`
`units: MB`
`type memory`
	Returns the maximum resident memory footprint
	Data Precision: 0
	A smaller number is better when doing HTML comparisons. inheritance_enabled 0

`flow.realtime`
`units: s`
`type time`
	Returns the real time since the last snapshot.
	Data Precision: 0
	A smaller number is better when doing HTML comparisons.

`flow.realtime.not_child`
`units: s`
`type time`
	Returns the real time since the last snapshot excluding child snapshots
	Data Precision: 0
	A smaller number is better when doing HTML comparisons.

`flow.realtime.total`
`units: s`
`type time`
	Returns the total real time
	Data Precision: 0
	A smaller number is better when doing HTML comparisons.

`flow.root_config` `type dict`	Returns the non-default root config option values Data Precision: 0
`flow.run_directory tcl`	Returns the password type directory from the current working directory. Data Precision: 0

Frequency Metrics

The following table provides the list of various frequency metrics.

Metric Object Name	Description
`frequency.violations`	Total number of violation reported Data Precision: 0 A smaller number is better when doing HTML comparison.
`frequency.violations.analysis_view:<view>`	Total number of violation reported per view Data Precision: 0 A smaller number is better when doing HTML comparison.

`frequency.violations.clock`	Total number of violation reported on clock net Data Precision: 0 A smaller number is better when doing HTML comparison.
`frequency.violations.clock.analysis_view:<view>`	Total number of violation reported on clock net per view Data Precision: 0 A smaller number is better when doing HTML comparison.

Power Metrics

The following table provides the list of various power metrics.

Metric Object Name	Description
`power` `units: mW`	Total power Data Precision: 2 A smaller number is better when doing HTML comparison.
`power.block.<block>` `units: mW`	Total per block power Data Precision: 2 A smaller number is better when doing HTML comparison.
`power.clock` `units: mW`	Total clock power Data Precision: 2 A smaller number is better when doing HTML comparison.
`power.clock.total` `units: mW`	Total clock power Data Precision: 2 A smaller number is better when doing HTML comparison.
`power.dynamic` `units: mW`	Total dynamic power Data Precision: 2 A smaller number is better when doing HTML comparison.

`power.dynamic.block` `units: mW`	Macro dynamic power Data Precision: 2 A smaller number is better when doing HTML comparison.
`power.dynamic.block.<block>` `units: mW`	Total per block dynamic power Data Precision: 2 A smaller number is better when doing HTML comparison.
`power.dynamic.clock` `units: mw`	Clock dynamic power Data Precision: 2 A smaller number is better when doing HTML comparison.
`power.dynamic.combinatorial` `units: mW`	Combinatorial dynamic power Data Precision: 2 A smaller number is better when doing HTML comparison.
`power.dynamic.hinst:<block>` `units: mW`	Total per block dynamic power Data Precision: 2 A smaller number is better when doing HTML comparison.
`power.dynamic.io u` `nits mW`	IO dynamic power Data Precision: 2 A smaller number is better when doing HTML comparison.
`power.dynamic.macro` `units: mW`	Macro dynamic power Data Precision: 2 A smaller number is better when doing HTML comparison.
`power.dynamic.sequential` `units: mW`	Sequential dynamic power Data Precision: 2 A smaller number is better when doing HTML comparison.

`power.dynamic.total` `units: mW`	Total dynamic power Data Precision: 2 A smaller number is better when doing HTML comparison.

`power.dynamic.`
`type: <combinatorial|clock_combinatorial|io|clock_sequential|sequential>`
`units: mW`
	Per type dynamic power
	Data Precision: 2
	A smaller number is better when doing HTML comparison.

`power.hinst:<block>` `units: mW`	Total per block power Data Precision: 2 A smaller number is better when doing HTML comparison.
`power.internal` `units: mW`	Total internal power Data Precision: 2 A smaller number is better when doing HTML comparison.
`power.internal.block` `units: mW`	Internal Macro power Data Precision: 2 A smaller number is better when doing HTML comparison.
`power.internal.block.<block>` `units: mW`	Total per block internal power Data Precision: 2 A smaller number is better when doing HTML comparison.
`power.internal.clock` `units: mw`	Internal clock power Data Precision: 2 A smaller number is better when doing HTML comparison.
`power.internal.clock_combinatorial` `units: mW`	Internal combinatorial clock power Data Precision: 2 A smaller number is better when doing HTML comparison.
`power.internal.clock_sequential` `units: mW`	Internal sequential clock power Data Precision: 2 A smaller number is better when doing HTML comparison.

`power.internal.combinatorial` `units: mW`	Internal combinatorial power Data Precision: 2 A smaller number is better when doing HTML comparison.
`power.internal.hinst:<block>` `units: mW`	Total per block internal power Data Precision: 2 A smaller number is better when doing HTML comparison.
`power.internal.io` `units: mW`	Internal IO power Data Precision: 2 A smaller number is better when doing HTML comparison.
`power.internal.macro` `units: mW`	Internal Macro power Data Precision: 2 A smaller number is better when doing HTML comparison.
`power.internal.sequential` `units: mW`	Internal sequential power Data Precision: 2 A smaller number is better when doing HTML comparison.
`power.internal.total` `units: mW`	Total internal power Data Precision: 2 A smaller number is better when doing HTML comparison.

`power.internal.type:`
`<combinatorial|clock_combinatorial|io|clock_sequential|sequential>`
`units: mW`
	Per type internal power
	Data Precision: 2
	A smaller number is better when doing HTML comparison.

`power.leakage` `units: mW`	Total leakage power Data Precision: 2 A smaller number is better when doing HTML comparison.

`power.leakage.block` `units: mW`	Macro leakage power Data Precision: 2 A smaller number is better when doing HTML comparison.
`power.leakage.block.<block>` `units: mW`	Total per block leakage power Data Precision: 2 A smaller number is better when doing HTML comparison.
`power.leakage.clock` `units: mw`	Clock leakage power Data Precision: 2 A smaller number is better when doing HTML comparison.
`power.leakage.clock_combinatorial` `units: mW`	Combinatorial clock leakage power Data Precision: 2 A smaller number is better when doing HTML comparison.
`power.leakage.clock_sequential` `units: mW`	Sequential clock leakage power Data Precision: 2 A smaller number is better when doing HTML comparison.
`power.leakage.combinatorial` `units: mW`	Combinatorial leakage power Data Precision: 2 A smaller number is better when doing HTML comparison.
`power.leakage.hinst:<block>` `units: mW`	Total per block leakage power Data Precision: 2 A smaller number is better when doing HTML comparison.
`power.leakage.io` `units: mW`	IO leakage power Data Precision: 2 A smaller number is better when doing HTML comparison.

`power.leakage.macro` `units: mW`	Macro leakage power Data Precision: 2 A smaller number is better when doing HTML comparison.
`power.leakage.sequential` `units: mW`	Sequential leakage power Data Precision: 2 A smaller number is better when doing HTML comparison.
`power.leakage.total units: mW`	Total leakage power Data Precision: 2 A smaller number is better when doing HTML comparison.
`power.leakage.type: <combinatorial|clock_combinatorial|io|clock_sequential|sequential>` `units: mW`	Per type leakage power Data Precision: 2 A smaller number is better when doing HTML comparison.
`power.switching` `units: mW`	Total switching power Data Precision: 2 A smaller number is better when doing HTML comparison.
`power.switching.block` `units: mW`	Macro switching power Data Precision: 2 A smaller number is better when doing HTML comparison.
`power.switching.block.<block>` `units: mW`	Total per block switching power Data Precision: 2 A smaller number is better when doing HTML comparison.
`power.switching.clock` `units: mw`	Clock switching power Data Precision: 2 A smaller number is better when doing HTML comparison.

`power.switching.clock_combinatorial` `units: mW`	Combinatorial clock switching power Data Precision: 2 A smaller number is better when doing HTML comparison.
`power.switching.clock_sequential` `units: mW`	Sequential clock switching power Data Precision: 2 A smaller number is better when doing HTML comparison.
`power.switching.combinatorial` `units: mW`	Combinatorial switching power Data Precision: 2 A smaller number is better when doing HTML comparison.
`power.switching.hinst:<block>` `units: mW`	Total per block switching power Data Precision: 2 A smaller number is better when doing HTML comparison.
`power.switching.io units: mW`	IO switching power Data Precision: 2 A smaller number is better when doing HTML comparison.
`power.switching.macro` `units: mW`	Macro switching power Data Precision: 2 A smaller number is better when doing HTML comparison.
`power.switching.sequential` `units: mW`	Sequential switching power Data Precision: 2 A smaller number is better when doing HTML comparison.
`power.switching.total` `units: mW`	Total switching power Data Precision: 2 A smaller number is better when doing HTML comparison.

`power.switching.type: <combinatorial|clock_combinatorial|io|clock_sequential|sequential>` `units: mW`	Per type switching power Data Precision: 2 A smaller number is better when doing HTML comparison.
`power.total` `units: mW`	Total power Data Precision: 2 A smaller number is better when doing HTML comparison.
`power.view`	Power analysis view Data Precision: 0

Rail Metrics

The following table provides the list of various rail metrics.

Metric Object Name	
`rail.averagedemandcurrent.net:<net>` `units: mA`	Average of TC_SUM waveform for net Data Precision: 3
`rail.averagesupplycurrent.net:<net>` `units: mA`	Average of VC_SUM waveform for net Data Precision: 3
`rail.gridcap.net:<net>` `units: pF`	Grid cap for net Data Precision: 3
`rail.intrinsiccap.net:<net>` `units: pF`	Intrinsic cap for net Data Precision: 3
`rail.ir.dynamic.avg.net:<net>` `units: V`	Avg dynamic IR value Data Precision: 3 A smaller number is better when doing HTML comparison.
`rail.ir.dynamic.max.net:<net>` `units: V`	Max dynamic IR value Data Precision: 3 A smaller number is better when doing HTML comparison.
`rail.ir.dynamic.min.net:<net>` `units: V`	Min dynamic IR value Data Precision: 3 A smaller number is better when doing HTML comparison.

`rail.ir.dynamic.view`	Dynamic view name Data Precision: 0
`rail.ir.dynamic.violations.net:<net>`	Number of resistor violating dynamic IR limit value Data Precision: 0 A smaller number is better when doing HTML comparison.
`rail.ir.static.avg.net:<net>` `units: V`	Avg static IR value Data Precision: 3 A smaller number is better when doing HTML comparison.
`rail.ir.static.max.net:<net>` `units: V`	Max static IR value Data Precision: 3 A smaller number is better when doing HTML comparison.
`rail.ir.static.min.net:<net>` `units: V`	Min static IR value Data Precision: 3 A smaller number is better when doing HTML comparison.
`rail.ir.static.view`	Static view name Data Precision: 0
`rail.ir.static.violations.net:<net>`	Number of resistor violating static IR limit value Data Precision: 0 A smaller number is better when doing HTML comparison.
`rail.loadingcap.net:<net>` `units: pF`	Loading cap for net Data Precision: 3
`rail.peakdemandcurrent.net:<net>` `units: mA`	Peak of TC_SUM waveform for net Data Precision: 3
`rail.peaksupplycurrent.net:<net>` `units: mA`	Peak of VC_SUM waveform for net Data Precision: 3
`rail.poweremreport.net:<net>`	Path to the powerem report for the net Data Precision: 0
`rail.referencevoltage.net:<net>` `units: V`	Reference design voltage Data Precision: 3
`rail.rj.avg.net:<net>`	Avg RJ Data Precision: 3 A smaller number is better when doing HTML comparison.

`rail.rj.max.net:<net>`	Max RJ Data Precision: 3 A smaller number is better when doing HTML comparison.
`rail.rj.min.net:<net>`	Min RJ Data Precision: 3 A smaller number is better when doing HTML comparison.
`rail.rj.violations.net:<net>`	Total RJ violation (above EM limit) Data Precision: 0 A smaller number is better when doing HTML comparison.
`rail.rushcurrent.net:<net>` `units: mA`	Rush current for net Data Precision: 3
`rail.thresholdvoltage.net:<net>` `units: V`	Threshold used for rail analysis Data Precision: 3
`rail.totalcap.net:<net>` `units: pF`	Total caps for net Data Precision: 3
`rail.totalinstanceconnected.net:<net>`	Total number of instance connected to the net Data Precision: 0
`rail.totalpowerswitchesturnedon.net:<net>`	Total power switches turned on Data Precision: 0
`rail.wakeuptime.net:<net>` `units: ns`	Wake up time for net Data Precision: 3
`rail.worstircycle.net:<net>` `units: ns`	Worst case time <interval number*resolution> Data Precision: 3
`rail.worstivreport.net:<net>`	Path to net specific iv file Data Precision: 0
`rail.worstivreport.power_domain:<domain>`	Path to domain specific iv file Data Precision: 0

Route Metrics

The following table provides the list of various route metrics.

Metric Object Name	Description
`route.drc`	Total DRC count Data Precision: 0 A smaller number is better when doing HTML comparison.
`route.drc.<layer>`	DRC count per layer Data Precision: 0 A smaller number is better when doing HTML comparison.
`route.drc.<type>`	DRC count per type Data Precision: 0 A smaller number is better when doing HTML comparison.
`route.drc.layer:<layer>`	DRC count per layer Data Precision: 0 A smaller number is better when doing HTML comparison.
`route.drc.total`	Total DRC count Data Precision: 0 A smaller number is better when doing HTML comparison.
`route.drc.type:<type>`	DRC count per type Data Precision: 0 A smaller number is better when doing HTML comparison.
`route.layers`	Number of routing layers Data Precision: 0
`route.map.<layer>`	Congesion map per layer Data Precision: 0 internal 1
`route.map.all`	Congestion map for all layers Data Precision: 0 internal 1
`route.map.hor`	Congestion map for hor layers Data Precision: 0 internal 1

`route.map.layer:<layer>`	Congesion map per layer Data Precision: 0 internal 1
`route.map.ver`	Congestion map for ver layers Data Precision: 0 internal 1
`route.overcon`	Total GR overcon Data Precision: 0 A smaller number is better when doing HTML comparison.
`route.overcon.<layer>`	GR overcon per layer Data Precision: 0 A smaller number is better when doing HTML comparison.
`route.overcon.layer:<layer>`	GR overcon per layer Data Precision: 0 A smaller number is better when doing HTML comparison.
`route.overcon.total`	Total GR overcon Data Precision: 0 A smaller number is better when doing HTML comparison.
`route.overflow.horizontal`	Trial-route horizontal overflow Data Precision: 3 A smaller number is better when doing HTML comparison.
`route.overflow.vertical`	Trial-route vertical overflow Data Precision: 3 A smaller number is better when doing HTML comparison.
`route.pitch.<layer>` `units: um`	Routing pitch per layer Data Precision: 3
`route.pitch.layer:<layer>` `units: um`	Routing pitch per layer Data Precision: 3
`route.shielding.nets`	Number of shielded nets Data Precision: 0
`route.shielding.percentage`	Shielding percentage Data Precision: 0

`route.via`	Total number of vias Data Precision: 0 A smaller number is better when doing HTML comparison.
`route.via.layer:<layer>`	Total number of vias per layer Data Precision: 0 A smaller number is better when doing HTML comparison.
`route.via.multicut`	Total number of MC vias Data Precision: 0 A smaller number is better when doing HTML comparison.
`route.via.multicut.<layer>`	Number of MC vias per layer Data Precision: 0 A smaller number is better when doing HTML comparison.
`route.via.multicut.layer:<layer>`	Number of MC vias per layer Data Precision: 0 A smaller number is better when doing HTML comparison.
`route.via.multicut.percentage`	Percentage of MC vias Data Precision: 0 A smaller number is better when doing HTML comparison.
`route.via.multicut.total`	Total number of MC vias Data Precision: 0 A smaller number is better when doing HTML comparison.
`route.via.singlecut`	Total number of SC vias Data Precision: 0 A smaller number is better when doing HTML comparison.
`route.via.singlecut.<layer>`	Number of SC vias per layer Data Precision: 0 A smaller number is better when doing HTML comparison.
`route.via.singlecut.layer:<layer>`	Number of SC vias per layer Data Precision: 0 A smaller number is better when doing HTML comparison.
`route.via.singlecut.percentage`	Percentage of SC vias Data Precision: 0

`route.via.singlecut.total`	Total number of SC vias Data Precision: 0 A smaller number is better when doing HTML comparison.
`route.via.total`	Total number of vias Data Precision: 0 A smaller number is better when doing HTML comparison.
`route.via.total.<layer>`	Total number of vias per layer Data Precision: 0 A smaller number is better when doing HTML comparison.
`route.width.<layer>` `units: um`	Routing width per layer Data Precision: 3
`route.width.layer:<layer>` `units: um`	Routing width per layer Data Precision: 3
`route.wirelength` `units: um`	Total wirelength Data Precision: 0 A smaller number is better when doing HTML comparison.
`route.wirelength.<layer>` `units: um`	Wirelength per layer Data Precision: 0 A smaller number is better when doing HTML comparison.
`route.wirelength.layer:<layer>` `units: um`	Wirelength per layer Data Precision: 0 A smaller number is better when doing HTML comparison.
`route.wirelength.total` `units: um`	Total wirelength Data Precision: 0 A smaller number is better when doing HTML comparison.

Timing Metrics

The following table provides the list of various timing metrics.

Metric Object Name	Description
`timing.analysis_views.all`	All views of setup/hold timing Data Precision: 0

`timing.analysis_views.all.number`	The number of all views of setup/hold timing Data Precision: 0
`timing.drv.max_cap`	Total max capacitance violation Data Precision: 0 A smaller number is better when doing HTML comparison.
`timing.drv.max_cap.total`	Total max capacitance violation Data Precision: 0 A smaller number is better when doing HTML comparison.
`timing.drv.max_cap.worst`	Worst max capacitance violation Data Precision: 0 A smaller number is better when doing HTML comparison.
`timing.drv.max_fanout`	Total max fanout violations Data Precision: 0 A smaller number is better when doing HTML comparison.
`timing.drv.max_fanout.total`	Total max fanout violations Data Precision: 0 A smaller number is better when doing HTML comparison.
`timing.drv.max_fanout.worst`	Worst max fanout violation Data Precision: 0 A smaller number is better when doing HTML comparison.
`timing.drv.max_length`	Total max length violations Data Precision: 0 A smaller number is better when doing HTML comparison.
`timing.drv.max_length.total`	Total max length violations Data Precision: 0 A smaller number is better when doing HTML comparison.
`timing.drv.max_length.worst` `units: um`	Worst max length violation Data Precision: 3 A smaller number is better when doing HTML comparison.
`timing.drv.max_tran`	Total max transition violations Data Precision: 0 A smaller number is better when doing HTML comparison.

`timing.drv.max_tran.total`	Total max transition violations Data Precision: 0 A smaller number is better when doing HTML comparison.
`timing.drv.max_tran.worst` `units: ns`	Worst max transition violation Data Precision: 3 A smaller number is better when doing HTML comparison.
`timing.drv.report_file`	Location of the timing DRV report file Data Precision: 0
`timing.hold.analysis_views.active`	All active views of hold timing Data Precision: 0
`timing.hold.analysis_views.active.number`	The number of all active views of hold timing Data Precision: 0
`timing.hold.analysis_views.enabled`	All enabled views of hold timing Data Precision: 0
`timing.hold.analysis_views.enabled.number`	The number of all enabled views of hold timing Data Precision: 0
`timing.hold.feps`	Total number of hold failing end points Data Precision: 0 A smaller number is better when doing HTML comparison.
`timing.hold.feps.<pathgroup>`	Total number of hold failing end points for a specific path group Data Precision: 0 A smaller number is better when doing HTML comparison.
`timing.hold.feps.<view>`	Total number of hold failing end points for a specific view Data Precision: 0 A smaller number is better when doing HTML comparison.
`timing.hold.feps.analysis_view:<view>`	Total number of hold failing end points for a specific view Data Precision: 0 A smaller number is better when doing HTML comparison.
`timing.hold.feps.analysis_view:` `<view>.path_group:<group>`	Total number of hold failing end points for a specific view and path group Data Precision: 0 A smaller number is better when doing HTML comparison.

`timing.hold.feps.path_group:<group>`	Total number of hold failing end points for a specific path group Data Precision: 0 A smaller number is better when doing HTML comparison.
`timing.hold.histogram` `Type: histogram`	Hold TNS histogram Data Precision: 0
`timing.hold.histogram.views` `Type: histogram`	Hold TNS histogram (per view) Data Precision: 0
`timing.hold.paths` `Type: timing_data`	Hold timing path details Data Precision: 0
`timing.hold.paths.path_group:reg2reg` `Type: timing_data`	Hold timing path reg2reg details Data Precision: 0
`timing.hold.pba.histogram` `Type: histogram`	Hold TNS histogram Data Precision: 0
`timing.hold.pba.histogram.views` `Type: histogram`	Hold TNS histogram (per view) Data Precision: 0

`timing.hold.tns`
`units: ns`
`Type: slack`
	Total negative hold slack (all path groups)
	Data Precision: 0

`timing.hold.tns.<pathgroup>`
`units: ns`
`Type: slack`
	Total negative hold slack for a specific path group
	Data Precision: 0

`timing.hold.tns.<view>`
`units: ns`
`Type: slack`
	Total negative hold slack for a specific view
	Data Precision: 0

`timing.hold.tns.analysis_view:<view>`
`units: ns`
`Type: slack`
	Total negative hold slack for a specific view
	Data Precision: 0

`timing.hold.tns.analysis_view:`
`<view>.path_group:<group>`
`units: ns`
`Type: slack`
	Total negative hold slack for a specific view and path group
	Data Precision: 0

`timing.hold.tns.path_group:<group>`
`units: ns`
`Type: slack`
	Total negative hold slack for a specific path group
	Data Precision: 0

`timing.hold.wns`
`units: ns`
`Type: slack`
	Worst negative hold slack (all path groups)
	Data Precision: 3

`timing.hold.wns.<pathgroup>`
`units: ns`
`Type: slack`
	Worst negative hold slack for a specific path group
	Data Precision: 3

`timing.hold.wns.<view>`
`units: ns`
`Type: slack`
	Worst negative hold slack for a specific view
	Data Precision: 3

`timing.hold.wns.analysis_view:<view>`
`units: ns`
`Type: slack`
	Worst negative hold slack for a specific view
	Data Precision: 3

`timing.hold.wns.analysis_view:`
`<view>.path_group:<group>`
`units: ns`
`Type: slack`
	Worst negative hold slack for a specific view and path group
	Data Precision: 3

`timing.hold.wns.path_group:<group>`
`units: ns`
`Type: slack`
	Worst negative hold slack for a specific path group
	Data Precision: 3

`timing.hold.worst_path.capture_clock`	Capture clock of the worst hold path Data Precision: 0
`timing.hold.worst_path.freq` `units: GHz`	Frequency: 1/(capture period-slack) of the worst hold path A larger number is better when doing HTML comparison.
`timing.hold.worst_path.frequency` `units: GHz`	Frequency: 1/(capture period-slack) of the worst hold path Data Precision: 0 A larger number is better when doing HTML comparison.
`timing.hold.worst_path.launch_clock`	Launch clock of the worst hold path Data Precision: 0

`timing.hold.worst_path.view`	Analysis view of the worst hold path Data Precision: 0
`timing.setup.analysis_views.active`	All active views of setup timing Data Precision: 0
`timing.setup.analysis_views.active.number`	The number of all active views of setup timing Data Precision: 0
`timing.setup.analysis_views.enabled`	All enabled views of setup timing Data Precision: 0
`timing.setup.analysis_views.enabled.number`	The number of all enabled views of setup timing Data Precision: 0
`timing.setup.feps`	Total number of setup failing end points Data Precision: 0 A smaller number is better when doing HTML comparison.
`timing.setup.feps.<pathgroup>`	Total number of setup failing end points for a specific path group Data Precision: 0 A smaller number is better when doing HTML comparison.
`timing.setup.feps.<view>`	Total number of setup failing end points for a specific view Data Precision: 0 A smaller number is better when doing HTML comparison.
`timing.setup.feps.analysis_view:<view>`	Total number of setup failing end points for a specific view Data Precision: 0 A smaller number is better when doing HTML comparison.
`timing.setup.feps.analysis_view:<view>.path_group:<group>`	Total number of setup failing end points for a specific view and path group Data Precision: 0 A smaller number is better when doing HTML comparison.
`timing.setup.feps.path_group:<group>`	Total number of setup failing end points for a specific path group Data Precision: 0 A smaller number is better when doing HTML comparison.
`timing.setup.histogram` `Type: histogram`	Setup TNS histogram Data Precision: 0

`timing.setup.histogram.views` `Type: histogram`	Setup TNS histogram (per view) Data Precision: 0
`timing.setup.paths` `Type: timing_data`	Setup timing path details Data Precision: 0
`timing.setup.paths.path_group:reg2reg` `Type: timing_data`	Setup timing path reg2reg details Data Precision: 0
`timing.setup.pba.histogram t` `ype histogram`	Setup TNS histogram Data Precision: 0
`timing.setup.pba.histogram.views` `Type: histogram`	Setup TNS histogram (per view) Data Precision: 0
`timing.setup.place.feps`	Total number of setup failing end points according to timing inside placement engine Data Precision: 0 A smaller number is better when doing HTML comparison. internal 1
`timing.setup.place.tns` `units: ns`	Total negative setup slack according to timing inside placement engine Data Precision: 0 A larger number is better when doing HTML comparison. internal 1
`timing.setup.place.wns` `units: ns`	Worst negative setup slack according to timing inside placement engine Data Precision: 3 c mp moreBetterIgnorePositive internal 1

`timing.setup.tns`
`units: ns`
`Type: slack`
	Total negative setup slack (all path groups)
	Data Precision: 0
	A larger number is better when doing HTML comparison.

`timing.setup.tns.<pathgroup>`
`units: ns`
`Type: slack`
	Total negative setup slack for a specific path group
	Data Precision: 0
	A larger number is better when doing HTML comparison.

`timing.setup.tns.<view>`
`units: ns`
`Type: slack`
	Total negative setup slack for a specific view
	Data Precision: 0
	A larger number is better when doing HTML comparison.

`timing.setup.tns.analysis_view:<view>`
`units: ns`
`Type: slack`
	Total negative setup slack for a specific view
	Data Precision: 0
	A larger number is better when doing HTML comparison.

`timing.setup.tns.analysis_view:<view>.path_group:<group>`
`units: ns`
`Type: slack`
	Total negative setup slack for a specific view and path group
	Data Precision: 0
	A larger number is better when doing HTML comparison.

`timing.setup.tns.path_group:<group>`
`units: ns`
`Type: slack`
	Total negative setup slack for a specific path group
	Data Precision: 0
	A larger number is better when doing HTML comparison.

`timing.setup.wns`
`units: ns`
`Type: slack`
	Worst negative setup slack (all path groups)
	Data Precision: 3
	A number below the Target: is better when doing HTML comparison.
	Target: 0

`timing.setup.wns.<pathgroup>`
`units: ns`
`Type: slack`
	Worst negative setup slack for a specific path group
	Data Precision: 3
	A number below the Target: is better when doing HTML comparison.
	Target: 0

`timing.setup.wns.<view>`
`units: ns`
`Type: slack`
	Worst negative setup slack for a specific view
	Data Precision: 3
	A number below the Target: is better when doing HTML comparison.
	Target: 0

`timing.setup.wns.analysis_view:<view>`
`units: ns`
`Type: slack`
	Worst negative setup slack for a specific view
	Data Precision: 3
	A number below the Target: is better when doing HTML comparison.
	Target: 0

`timing.setup.wns.analysis_view:<view>.path_group:<group>`
`units: ns`
`Type: slack`
	Worst negative setup slack for a specific view and path group
	Data Precision: 3
	A number below the Target: is better when doing HTML comparison.
	Target: 0

`timing.setup.wns.path_group:<group>`
`units: ns`
`Type: slack`
	Worst negative setup slack for a specific path group
	Data Precision: 3
	A number below the Target: is better when doing HTML comparison.
	Target: 0

`timing.setup.worst_path.capture_clock`	Capture clock of the worst setup path Data Precision: 0
`timing.setup.worst_path.freq` `units: GHz`	Frequency: 1/(capture period-slack) of the worst setup path A larger number is better when doing HTML comparison.
`timing.setup.worst_path.frequency` `units: GHz`	Frequency: 1/(capture period-slack) of the worst setup path Data Precision: 0 A larger number is better when doing HTML comparison.
`timing.setup.worst_path.launch_clock`	Launch clock of the worst setup path Data Precision: 0
`timing.setup.worst_path.view`	Analysis view of the worst setup path Data Precision: 0
`timing.si.glitches`	Number of SI glitches Data Precision: 0 A smaller number is better when doing HTML comparison.
`timing.si.noise` `units: ns`	Amount of noise delay remaining Data Precision: 3 A smaller number is better when doing HTML comparison.

create_metric_page

```
create_metric_page
[-help]
[-after page_name]
[-before page_name]
[-format {html excel text}]
-name page_name
```

Creates a new metric page.

Creation of a metric page is the first step in creating a metric table for HTML display. After creating a metric page, you need to specify the heading for the metric table using the `create_metric_table_heading` command. After specifying the metric table heading, you create the metric table using the `create_metric_table` command.

Parameters

`-after page_name`	Specifies to insert after the page with this name. *Date_type*: string, optional
`-before page_name`	Specifies to insert before the page with this name. *Date_type*: string, optional
`-format {html excel text}`	Specifies configuration for the `report_metric` format. *Date_type*: enum, optional
`-help`	Prints out the command usage. For a detailed description of the command, use the man command: `man create_metric_page`
`-name page_name`	Specifies the name of the new page. The page name must be unique. *Date_type*: string, required

Examples

- The following command creates the metric page, `User`.

  ```
  create_metric_page -name User
  ```

  The following command creates the `Test` metric page after the `Summary` page.

  ```
  create_metric_page -after Summary -name Test
  ```

Related Information

- The Unified Metrics chapter in the Innovus Stylus Common UI User Guide
- `create_metric_table_heading`
- `create_metric_table`

create_metric_table

```
create_metric_table
[-help]
[-auto_hide]
[-collapsible_key]
[-format {html excel text}]
-heading table_heading
[-hide_footers]
[-hide_graph_footers]
[-hierarchy_separator separator]
[-id table_id]
[-key regexp_key]
[-per_snapshot]
[-show_label_every frequency]
-type {horizontal vertical cpu dict time_statement histogram imageplot}
```

Creates a metric table.

To create a metric table, you first need to create a metric page using the `create_metric_page` command. After creating a metric page, you need to specify the heading for the metric table using the `create_metric_table_heading` command. After specifying the metric table heading, you create the metric table using the `create_metric_table` command.

Parameters

`-auto_hide`	Specifies whether to auto hide the table. *Date_type*: bool, optional
`-collapsible_key`	Specifies to include collapse key columns in the table with the option to expand them. *Date_type*: bool, optional
`-format {html excel text}`	Specifies configuration for the `report_metric` format. *Date_type*: enum, optional

`-heading` *`heading_id`*	Specifies the id of the heading to append. You need to specify the table heading already created using the `create_metric_table_heading` command.
	Date_type: string, required
	For example, the following command gives an error as the metric table heading `statistics` has not been defined before specifying the heading while creating the table.
	`@[DEV]innovus 5> create_metric_table -id table_id -heading statistics -type horizontal`
	`heading id {statistics} not found`
	`@[DEV]innovus 6> create_metric_table -heading statistics -type horizontal`
	`heading id {statistics} not found`
	`@[DEV]innovus 7>`

`-hide_footers`	Hides the minimum, average, and maximum rows from the bottom of the table. Graph buttons are still shown. *Date_type*: bool, optional
`-hide_graph_footers`	Hides the graph buttons from the bottom of the table. *Date_type*: bool, optional
`-hierarchy_separator` *`separator`*	Shows the metric names with hierarchy using the provided separator. *Date_type*: string, optional
`-id` *`table_id`*	Specifies the id for the new table. *Date_type*: string, optional
`-key` *`regexp_key`*	Specifies the key for any regexp. *Date_type*: string, optional
`-per_snapshot`	Specifies to expand the table per snapshot. *Date_type*: bool, optional
`-show_label_every` *`frequency`*	Specifies the frequency of X-axis labels in the histogram. *Date_type*: int, optional
`-type {horizontal vertical cpu dict time_statement histogram imageplot}`	Specifies the type (horizontal vertical cpu dict time_statement histogram imageplot) of table. *Date_type*: enum, required

Example

The following commands create the `table_s` metric table:

```
innovus1> create_metric_page -name User
```

```
innovus2> create_metric_table_heading -id statistics -page User \
  -title "Average Timing Values Per Snapshot"
innovus3> create_metric_table  -id table_s -heading statistics \
  -per_snapshot -type  horizontal
```

Related Information

- The Unified Metrics chapter in the Innovus Stylus Common UI User Guide
- `create_metric_table_heading`
- `report_metric`

create_metric_table_heading

```
create_metric_table_heading
[-help]
[-format {html excel text}]
[-id heading_id]
-page page_name
-title heading_title
[-type {fixed collapse}]
```

Create a new metric heading.

You create the heading for the metric page you created using the `create_metric_page`. After specifying the metric table heading, you create the metric table using the `create_metric_table` command.

Parameters

-help	Prints out the command usage. For a detailed description of the command, use the man command: `man create_metric_table_heading`
`-format {html excel text}`	Specifies configuration for the `report_metric` format. *Date_type*: enum, optional
`-id heading_id`	Specifies the id for the new heading. *Date_type*: string, optional
`-page page_name`	Specifies the page name where the heading will be added. *Date_type*: string, required
`-title heading_title`	Specifies the title of the heading. *Date_type*: string, required
`-type {fixed collapse}`	Specifies the type of heading element, fixed or collapse. *Date_type*: enum, optional

Example

The following commands create the `table_s` metric table with the heading `Average Timing Values Per Snapshot`:

```
innovus2> create_metric_table_heading -id statistics -page User \
  -title "Average Timing Values Per Snapshot"
innovus3> create_metric_table  -id table_s -heading statistics \
  -per_snapshot -type  horizontal
```

Related Information

- The Unified Metrics chapter in the Innovus Stylus Common UI User Guide
- `create_metric_page`
- `create_metric_table`

create_snapshot

```
create_snapshot
[pattern]
[-auto {min default}]
-categories [* flow design setup power check clock hold route]
[-exclude_time_metric]
[-name snapshot_name]
```

Creates snapshots of the current metric list at various points in the flow.

All super commands that save metrics call `create_snapshot` to save all metrics in a consistent manner. When `create_snapshot` is called, it creates an internal record designated by a label. The label for super commands are the command names.

Metrics that do not require extraction and/or timing/power analysis are calculated when the metric snapshot is saved. Timing/clock/power metrics are reported only if they are up-to-date (i.e. if `time_design`/`report_ccopt*`/`report_power` have been run). Metrics are saved as part of db and can be accessed using `get_db`. Reporting commands calculate metrics and push them into the metric stream. However, these metrics remain pending until you run `create_snapshot`. This allows the snapshot to contain metrics from more than one reporting command.

Note: Metric values should be saved such that the unit is easily known/accessible.

Parameters

`-auto {min default}`	Specifies the set of auto metrics to `run`. *Date_type*: enum, optional
`-categories [* flow design setup power check clock hold route]`	Specifies the metric categories to compute. *Date_type*: string, optional
`-exclude_time_metric`	Excludes the CPU and wall time from the parent snapshot calculations. *Date_type*: bool, optional
`-name snapshot_name`	Specifies the name of the snapshot. *Date_type*: string, optional
`-help`	Outputs a brief description that includes type and default information for each `create_snapshot` parameter. For a detailed description of the command and all of its parameters, use the man command: `man create_snapshot`
pattern	Specifies specific metrics to save. *Date_type*: string, optional

Example

In the following example, the metrics calculated by the reporting commands. `report_power` and `report_timing_summary`, are pushed into the metric stream by the `create_snapshot` command:

```
innovus1> report_power

innovus2> report_timing_summary

innovus3> create_snapshot -name "setup_and_power"
```

Related Information

- `get_db`
- `report_power`
- `report_timing_summary`
- `time_design`
- The Unified Metrics chapter in the Innovus Stylus Common UI User Guide

define_metric

```
define_metric
[-help]
[-compare {more_better less_better more_better_below_target less_better_above_target
more_better_ignore_positive}]
[-description metricDescription]
[{-id databaseID}]
[-inheritance_tcl TclName]
-name metric_name
[-no_inheritance]
[-precision number_of_decimal_places]
[-target metric_name]
[{-tcl TCL_command_list}[-tools listOfTools] ]
[-threshold reporting_unit]
[-type metricType]
[-units reporting_unit]
[-warning threshold]
```

Defines new metrics that are required by the user.

You can use this command to create new metrics that you need to track. To create new metrics, you need to provide the TCL code required to generate the metric. The TCL can be a sequence of commands that return a value or a procedure that returns a value.

The following command provides the TCL commands `.base_cell.name == HDR*` to define the design.switch_cells metric.

```
define_metric -name design.switch_cells -tcl { return [get_db insts -if {.base_cell.name == HDR*}] }
```

Similarly, a proc can be used to define a metric. The following command provides the `my_get_switch_cells{}` proc to define the `design.switch_cells` metric.

```
define_metric -name design.switch_cells -tcl { return [my_get_switch_cells{}] }
```

The metric is calculated and stored with the other metrics when `create_snapshot` is executed. All `define_metric` commands are saved and restored in the design database.

Parameters

`-compare {more_better less_better more_better_below_target less_better_above_target more_better_ignore_positive}`	Specifies the compare function, `{more_better less_better more_better_below_target less_better_above_target more_better_ignore_positive}`. The comparison options show the good or bad color when the value is below or above the target value as appropriate. *Data_type*: enum, optional
`-description metricDescription`	Specifies the description of the metric. *Data_type*: string, optional

`-id` *databaseID*	Specifies the database ID. *Data_type*: string, optional
`-inheritance_tcl` *TclName*	Specifies Tcl for combining children values. The metric is not executed in the global namespace. The equivalent proc args is {children}. *Data_type*: string, optional
`-name` *metric_name*	Specifies the name of the metric. *Data_type*: string, required
`-no_inheritance`	Disables the inheritance of the metric. *Data_type*: bool, optional
`-help`	Outputs a brief description that includes type and default information for each `define_metric` parameter. For a detailed description of the command and all of its parameters, use the man command: `man define_metric`
`-precision` *number_of_decimal_places*	Specifies the number of decimal places. *Default*: `0` *Data_type*: int, optional
`-target` *metric_name*	Specifies the target or the name of metric containing the target value. *Data_type*: string, optional
`-tcl { TCL_command_list }`	Specifies the list of TCL commands or proc to generate (return) the metric which is of the format `{<value> <unit>}`. When the `-tcl` option is specified, the TCL associated with the metric will be executed whenever `create_snapshot` is called, and the metric will be captured as part of the snapshot. *Data_type*: string, optional
`-threshold` *reporting_unit*	Specifies the threshold for comparing numbers in HTML. If the delta between the two numbers being compared is less than the threshold then the metric will be colored “yellow”. If it is outside the threshold, then it is colored red or green depending on the compare function i.e. whether more is better or less is better. *Data_type*: string, optional
`-tools` *listOfTools*	Specifies the list of tools which the tcl code will run. *Data_type*: string, optional
`-type` *metricType*	Specifies the type of metric.

Data_type: string, optional

The various metrics types are:

- imageplot: A metric type that the renders to an image in the HTML report.
  - Format:

```
<name1> { \
 fill_color "<color>" \
  stroke_color "<color>"\
  data <list of data sets>
} \
<name2> { \
<sub-name1> { \
    fill_color <color> \
    data <list of data sets> \
  } \
  <sub-name2> {...} \
}
```

  Where a data set can be one or more of the following:

  The data on each layer is a string of coordinates (separated by sequences of whitespace and comma) with types. These types are:

  - R - rectangle (x, y, width, height)
  - B - bounding box (x_min, y_min, x_max, y_max)
  - G - closed polygon (x, y, x, y, ...)
  - L - open polyline (x, y, x, y, ...)
  - C - circle (x, y, r)
  - P - point (x, y)

  For rectangles, boxes, circles and points, multiple shapes can be defined without having to repeat the type letter. e.g. R1,2,3,4,5,6,7,8 is two rectangles.

  A value can be defined for shapes by adding a V suffix to the type; this value can then be used to color the shapes according to the value for heat maps. The value is always the first number after the type. This means that two points with values 20 and 40 may look like: PV20,1,2 40,6,5 or: PV20,1,2PV40,6,5.

  Example imageplot TCL dict data:

```
macros { \
fill_color "#FF0000" \
  stroke_color "#0000FF88" \
  data "R5,8,1,2 6,9,2,1 G1,2,3,4,5,6,7,8" \
} \
drcs { \
placement { \
    fill_color "#FF0000" \
    data "P3,4" \
  } \
  routing {...} \
}
```

Layers in the image are named (macros, congestion, normal, other in the above example) and support hierarchy. Layers should not be nested beyond one level deep. The hierarchy is used in the renderer (see below) to toggle the visibility of the layers. It is an error for any layer to be called data (reserved to keep hierarchy clear).

Colors can be defined either as hex or RGB values with optional opacity. Hex values are prefixed with #. RGB values are separated with whitespace.

- #FF0000: Red
- #00FF0040: Green (25% opaque)
- 0 0 255: Blue
- 255 0 255 192: Magenta (75% opaque)

- histogram: The data will be structured as a set of series with shared X-axis labels that are rendered as a stacked histogram.

  Format (TCL dict):

```
labels <list of labels> \
axis_x <name> \
axis_y <name> \
series { \
<series1> { \
   values <list of buckets \
   color "#FF0000" \
} \
```

```
<series2> { \
   values {2 1 0 0} \
   color "#0000FF" \
} \
}
```

Example:

```
get_metric timing.setup.histogram
timing.setup.histogram {labels {-0.320 -0.240 -0.160
-0.080} \
                        series { \
                          reg2reg {color #FF0000
values {43 3 2 3 0}} \
                          reg2cgate {color #0000FF
values {22 4 5 0 0}} \
                        } \
                        axis_x WNS(ns) \
                        axis_y {Failing End Points} \
                        }
```

The labels (a list of strings, shown in the order provided) define the X-axis labels for the histogram, the values in each series define the Y-axis values.

Colors can be defined either as hex or RGB values with optional opacity. Hex values are prefixed with #. RGB values are separated with whitespace.

```
#FF0000       - Red
0 0 255       - Blue
```

- html: Supports the inclusion of raw HTML in a metric. You can place simple html into tables allowing simple links to reports. For example, the following command defines `test_html` as an html type of metric:

  ```
  define_metric -name test_html -type html
  ```

  You can set the html metric, test_html, using the `set_metric` command:

  ```
  set_metric -name test_html -value {<a
  href="url">test</a>}
  ```

- radar: Helps to identify the characteristics of each run by a single graph. It is useful in cases where histogram cannot be used for comparing several runs.
  If a series type is radar, then they must all be radar - i.e. there

	is no mixing of other graph types with radar. Different number of labels/values will result in different points in the graph but the number of labels must match the number of values and all series must have the same number of values.
	
	`{ labels {RCI Fmax Area Power DFM} \`
	`series { \`
	`scenarioA { \`
	`color #0000FF \`
	`values {32 32 28 12 15} \`
	`type radar\`
	`} \`
	`senarioB { \`
	`color #FFA500 \`
	`values {12 12 12 21 28}`
	`type radar\`
	`} \`
	`} \`
	`}`

`-units` *reporting_unit*	Specifies the unit for reporting. *Data_type*: string, optional
`-warning` *threshold*	Specifies the warning threshold or name of metric containing the threshold value. *Data_type*: string, optional

Examples

- The following command defines the the `X20s` metric:

  ```
  define_metric -name X20s
  set_metric -name X20s \
  -value [get_db insts -expr {[regexp X20 $obj(.base_cell)]}]
  ```

  Alternatively, you can define the X20s metrics as follows:

  ```
  define_metric -name X20s -tcl {\
  return [get_db insts -expr {[regexp X20 $obj(.base_cell)]]} \
  }
  ```

  With the `-tcl` option, the TCL associated with the X20s metric will be executed whenever create_snapshot is called.

Related Information

- The Unified Metrics chapter in the Innovus Stylus Common UI User Guide
- `create_snapshot`

delete_metric

```
delete_metric
-help
[{-id databaseID}]
[-name metric_name]
```

Deletes a previously defined (user) metric from all labels.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `delete_metric` parameter. For a detailed description of the command and all of its parameters, use the man command: `man delete_metric`
`-id databaseID`	Specifies the database ID. *Data_type*: string, optional
`-name metric_name`	Name of the metric to delete. *Data_type*: string, optional

Examples

- The following command deletes the `design.vtmix` metric:

```
delete_metric -name design.vtmix
```

Related Information

- `define_metric`
- `update_metric`
- The Unified Metrics chapter in the Innovus Stylus Common UI User Guide

delete_metric_page

```
delete_metric_page
[-help]
[-format {html excel text}]
-name page_name
```

Deletes a metric page.

Parameters

`-help`	Prints out the command usage. For a detailed description of the command, use the man command: `man delete_metric_page`
`-format {html excel text}`	Specifies configuration for the `report_metric` format. *Data_type*: enum, optional
`-name` *page_name*	Specifies the name of the metric page to be deleted. *Data_type*: string, required

Example

- The following command creates the metric page, `User`.

  ```
  delete_metric_page -name User
  ```

Related Information

- `create_metric_page`
- The Unified Metrics chapter in the Innovus Stylus Common UI User Guide

delete_metric_run

```
delete_metric_run
-help
[-name metric_name]
```

Deletes the specified metric run.

It deletes a set of metrics loaded using `read_metric`.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `delete_metric_run` parameter. For a detailed description of the command and all of its parameters, use the man command: `man delete_metric_run`
`-name` *`metric_name`*	Name of the metric run to be deleted. It defaults to the "current" run. This parameter corresponds to the `-id` parameter of the `read_metric` command. *Data_type*: string, optional

Example

- The following command deletes `run1`, which is read by `read_metric`.

```
read_metric -id run1 metrics.json
delete_metric_run -name run1
```

Related Information

- `read_metric`
- The Unified Metrics chapter in the Innovus Stylus Common UI User Guide

delete_metric_table

```
delete_metric_table
[-help]
[-format {html excel text}]
[-id table_id]
```

Deletes a metric table.

Parameters

`-help`	Prints out the command usage. For a detailed description of the command, use the man command: `man delete_metric_table`
`-format {html excel text}`	Specifies configuration for the `report_metric` format. *Data_type*: enum, optional
`-id table_id`	Specifies the id for the table to be deleted. *Data_type*: string, optional

Related Information

- `report_metric`
- The Unified Metrics chapter in the *Innovus Stylus Common UI User Guide*

delete_metric_table_heading

```
delete_metric_table_heading
[-help]
[-format {html excel text}]
[-id heading_id]
```

Deletes a metric heading.

Parameters

-help	Prints out the command usage. For a detailed description of the command, use the man command: `man delete_metric_table_heading`
`-format {html excel text}`	Specifies configuration for the specified `report_metric` format. *Data_type*: enum, optional
`-id` *heading_id*	Specifies the id of the heading to be deleted. *Data_type*: string, optional

Related Information

- `report_metric`
- The Unified Metrics chapter in the *Innovus Stylus Common UI User Guide*

get_metric

```
get_metric
-help
pattern_name
[-branch list_of_patterns]
[-exclude_inherited]
[-id id_name]
[-names label_name]
[-pending]
[-raw]
```

Gets the current state of metric(s).

You can use this command to query the current state of any metric or metrics from the current specified list of metrics as well as a previous state.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `get_metric` parameter. For a detailed description of the command and all of its parameters, use the man command: `man get_metric`
`-branch list_of_patterns`	Specifies list of patterns to match snapshot branches. This will limit all output to metrics stored in snapshots whose branch matches the specified branch. This requires any inheritance to be re-run to exclude inherited values from non-matching snapshots. *Data_type*: string, optional
`-exclude_inherited`	Specifies not to return any inherited value. *Data_type*: bool, optional
`-id id_name`	Specifies the database ID to output. Default is current database. *Data_type*: string, optional
`-names label_name`	List of patterns to match snapshot names. *Data_type*: string, optional
`pattern_name`	List of patterns to match metrics. This option supports wildcards. *Data_type*: string, optional
`-pending`	Returns metrics from the pending (not yet created) snapshot. *Data_type*: bool, optional
`-raw`	Returns metrics as a raw value i.e without normalization to the defined units. *Data_type*: bool, optional

Examples

- The following command gets the worst negative setup slack for all path groups for the preCTS snapshot from the gold database.

```
innovus1> get_metric -id gold -names prects timing.setup.wns

prects {timing.setup.wns -0.042}
```

- The following command gets the total number of instances for the sync_opt and prects snapshots from the gold database.

```
innovus2 > get_metric -id gold -names "syn_opt prects" design.instances

syn_opt {design.instances 658410} prects {design.instances 708621}
```

Related Information

- The Unified Metrics chapter in the Innovus Stylus Common UI User Guide

get_metric_alias

```
get_metric_alias
[-help]
```

Returns a tcl dictionary of aliases as dictionaries along with the from, to, and order properties.

Parameters

None

Example

The following command renames `path_group:reg2reg` to `reg2reg` with priority order `1`.

```
> set_metric_alias -from path_group:reg2reg -to reg2reg -order 1
```

The following command renames `path_group:reg2cgate` to `reg2cgate` with priority order `2`.

```
> set_metric_alias -from path_group:reg2cgate -to reg2cgate -order 2
```

The following command queries the set of aliases:

```
> get_metric_alias
```

Output

```
>path_group:reg2reg {to reg2reg order 1} path_group:reg2cgate {to reg2cgate order 2}
```

Related Information

- The Unified Metrics chapter in the *Innovus Stylus Common UI User Guide*

get_metric_config

```
get_metric_config
-help
-format {html excel text}
```

Gets the current metric configuration setting.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `get_metric_config` parameter. For a detailed description of the command and all of its parameters, use the man command: `man get_metric_config`
`-format {html excel text}`	Specifies configuration for the specified `report_metric` format. *Data_type*: enum, optional

Example

The following prints the current defined pages, heading and tables:

```
foreach page_name [dict keys [get_metric_config ]] {
  puts "\nPage : $page_name"
  puts "-----------------------------------------"
  foreach page [dict get [get_metric_config] $page_name] {
    foreach key [dict keys $page] {
      if {($key == "id")} {
        set id [dict get $page $key]
      }
      if {($key == "component")} {
         puts "$key : $id ([dict get $page $key])"
      }
    }
  }
}
```

Related Information

- `report_metric`
- The Unified Metrics chapter in the *Innovus Stylus Common UI User Guide*

get_metric_definition

```
get_metric_definition
-help
[-id metric_id]
-name metric_name
```

Returns the metric definition of the specified metric.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `get_metric_definition` parameter. For a detailed description of the command and all of its parameters, use the man command: `man get_metric_definition`
`-id metric_id`	Specifies the database ID to output. *Data_type*: string, optional
`-name label_name`	Specifies the metric name. *Data_type*: string, required

Examples

- The following command queries the attributes of the `timing.setup.wns` metric:

```
innovus 1> get_metric_definition -name timing.setup.wns

metric timing.setup.wns

units ns

type slack

description {Worst negative setup slack (all path groups)}

dp 3

cmp moreBetterBelowTarget

target 0
```

- The following command queries all defined metrics:

```
innovus 1> foreach m [get_metric_definition -name *] {puts $m }
```

Related Information

- The Unified Metrics chapter in the *Innovus Stylus Common UI User Guide*

get_metric_header

```
get_metric_header
-help
[-id db_id]
-name header_name
```

Returns the metric header information set by the `set_metric_header` command.

Parameters

`-id` *db_id*	Specifies the database ID. *Data_type*: string, optional
`-name` *header_name*	Specifies the name of the header. *Data_type*: string, required
`-help`	Outputs a brief description that includes type and default information for each `get_metric_header` parameter. For a detailed description of the command and all of its parameters, use the man command: `man get_metric_header`

Example

The following command sets the value `bar` of the metric header `foo`:

```
> set_metric_header -name timing -value high
```

The following command returns the header value of the metric header `foo`

```
> get_metric_header -name timing

high
```

Related Information

- The Unified Metrics chapter in the *Innovus Stylus Common UI User Guide*
- `set_metric_header`

metric Category Attributes

The root attributes in the metric category are:

- metric_advanced_url_endpoint
- metric_capture_depth
- metric_capture_design_image
- metric_capture_design_image_blockages
- metric_capture_design_image_power_intent
- metric_capture_design_image_route_drc
- metric_capture_max_drc_markers
- metric_capture_min_count
- metric_capture_overwrite
- metric_capture_pba_tns_histogram
- metric_capture_timing_analysis_mode
- metric_capture_per_view
- metric_capture_reg2reg_metrics
- metric_capture_timing_paths
- metric_capture_timing_path_groups
- metric_capture_tns_histogram
- metric_capture_tns_histogram_buckets
- metric_capture_tns_histogram_paths
- metric_capture_tns_histogram_max_slack
- metric_capture_vth_per_power_domain
- metric_category_default
- metric_current_run_id
- metric_enable
- metric_summary_metrics

To set the value of any root attribute, use the `set_db` command:

```
set_db attribute_name value
```

To get the current value of any root attribute, use the `get_db` command:

```
get_db attribute_name
```

To get the current value of all attributes in the `metric` category, use the following command:

```
get_db -category metric
```

metric_advanced_url_endpoint

```
metric_advanced_url_endpoint base_url
```

Default: `""`

Data_type: string, read/write

Specifies the base URL for the advanced metric server.

metric_capture_depth

```
metric_capture_depth depth
```

Default: `1`

Data_type: int, read/write

Specifies the depth for capturing hinst design and power metrics.

metric_capture_design_image

```
metric_capture_design_image {true | false}
```

Default: `true`

Data_type: bool, read/write

Captures design image.

metric_capture_design_image_blockages

```
metric_capture_design_image_blockages {true | false}
```

Default: `true`

Data_type: string, read/write

Captures blockages in the design image.

metric_capture_design_image_power_intent

```
metric_capture_design_image_power_intent {true | false}
```

Default: `true`

Data_type: string, read/write

Captures power intent objects in the design image.

metric_capture_design_image_route_drc

```
metric_capture_design_image_route_drc {true | false}
```

Default: `true`

Data_type: string, read/write

Captures route DRC markers in the design image.

metric_capture_max_drc_markers

`metric_capture_max_drc_markers` *max_number*

Default: `1000`

Data_type: int, read/write

Specifies the maximum number of DRC markers to include in metric image.

metric_capture_min_count

`metric_capture_min_count` *instanceCount*

Default: `1000`

Data_type: int, read/write

Specifies the minimum instance count for capturing hinst design and power metrics.

metric_capture_overwrite

`metric_capture_overwrite {true | false}`

Default: `false`

Data_type: bool, read/write

Overwrites the pending metrics during the `create_snapshot` category capture.

metric_capture_pba_tns_histogram

`metric_capture_pba_tns_histogram {true | false}`

Default: `false`

Data_type: bool, read/write

Captures the PBA histogram data for TNS.

metric_capture_timing_analysis_mode

`metric_capture_timing_analysis_mode {gba | ipba}`

Default: `gba`

Data_type: enum, read only

Controls the timing analysis mode for metric capture during create_snapshot categories setup and hold; gba, ipba.

- `gba`: graph based analysis
- `ipba`: path based analysis, retime_method == path_slew_propagation, retime_mode == exhaustive, timing_path_based_exhaustive_pba_bounded_mode == true

metric_capture_per_view

`metric_capture_per_view {true | false}`

Default: `true`

Data_type: bool, read/write

Captures the timing metrics per analysis_view.

metric_capture_reg2reg_metrics

`metric_capture_reg2reg_metrics [true | false}`

Default: `false`

Data_type: bool, read/write

Forces the capture of reg2reg metrics regardless of path groups.

metric_capture_timing_paths

`metric_capture_timing_paths` *`numberOfPaths`*

Default: `10`

Data_type: int, read/write

Specifies the number of paths to capture for detailed display.

metric_capture_timing_path_groups

`metric_capture_timing_path_groups` *`path_group`*

Default: `10`

Data_type: string, read/write

Controls the path groups for which detailed timing path information (full timing path report and display) is to be captured . By default, it is empty and the tool captures all and reg2reg. If the attribute is set to a list of groups, the tool will use the listed groups. The related metric name is `timing.setup.paths.path_group:<group>`.

metric_capture_tns_histogram

`metric_capture_tns_histogram {true | false}`

Default: `true`

Data_type: bool, read/write

Captures the histogram data for TNS.

metric_capture_tns_histogram_buckets

`metric_capture_tns_histogram_buckets` *`numberOfBuckets`*

Default: `50`

Data_type: int, read/write

Specifies the number of buckets for the TNS histogram.

metric_capture_tns_histogram_paths

`metric_capture_tns_histogram_paths` *`numberOfPaths`*

Default: `10000`

Data_type: int, read/write

Specifies the number of paths to capture for the TNS histogram.

metric_capture_tns_histogram_max_slack

`metric_capture_tns_histogram_max_slack` *`maximum_slack_value`*

Default: `0`

Data_type: int, read/write

Specifies the maximum slack value for the TNS histogram. You can change its value if you need to shift the y axis.

metric_capture_vth_per_power_domain

`metric_capture_vth_per_power_domain {true | false}`

Default: `false`

Data_type: bool, read/write

Captures vth metrics per power domain.

metric_category_default

`metric_category_default` *`category`*

Default: `design`

Data_type: string, read/write

Specifies to capture the default metric categories if none are provided.

metric_current_run_id

```
metric_current_run_id
```

Default: `""`

Data_type: string, read/write

Specifies the current unique run ID returned by the advanced metric server.

metric_enable

```
metric_enable {true | false}
```

Default: `false`

Data_type: bool, read/write

Enables metric snapshot capture with `create_snapshot`.

metric_summary_metrics

```
metric_summary_metrics {flow.cputime flow.realtime timing.setup.tns timing.setup.wns}
```

Default: flow.cputime flow.realtime timing.setup.tns timing.setup.wns

Data_type: string, read/write

Specifies summary metrics to be inherited when viewing snapshots and runs.

pop_snapshot_stack

```
pop_snapshot_stack
```

Pops the metrics saved since the last push.

The `pop_snapshot_stack` command works in conjunction with `push_snapshot_stack` to create metric hierarchies. Any metrics captured inside a push/pop must be saved via `create_snapshot` (implicit or explicit) or they will be lost.

Parameters

None

Example

The following code shows the usage of `pop_snapshot_stack`:

```
read_design ...
push_snapshot_stack
```

```
place_design
pop_snapshot_stack
create_snapshot -label place
push_snapshot_stack
opt_design -pre_cts
pop_snapshot_stack
create_snapshot -label prects
push_snapshot_stack
ccopt_design
time_design -post_cts
time_design -post_cts -hold
pop_snapshot_stack
create_snapshot -label cts
# ------------------------------------------------------------------
```

This will result in the following snapshot hierarchy:

```
place {
place_design {
<metrics>
}
}
prects {
opt_design -pre_cts {
<metrics>
}
}
cts {
ccopt_design {
<metrics>
}
time_design -post_cts {
<metrics>
}
time_design -post_cts -hold {
<metrics>
}
}
```

Related Information

- create_snapshot
- push_snapshot_stack
- The Unified Metrics chapter in the *Innovus Stylus Common UI User Guide*

push_snapshot_stack

`push_snapshot_stack`

Creates a hierarchy for metric capture. All metrics captured inside the hierarchy is saved at this level during snapshot creation (implicit or explicit). Snapshot hierarchies can be nested so any number of hierarchies are supported.

Note: Each `push_snapshot_stack` must have a corresponding `pop_snapshot_stack`.

Parameters

None

Example

The following code shows the usage of `push_snapshot_stack`:

```
read_design ...
push_snapshot_stack
place_design
pop_snapshot_stack
create_snapshot -label place
push_snapshot_stack
opt_design -pre_cts
pop_snapshot_stack
create_snapshot -label prects
push_snapshot_stack
ccopt_design
time_design -post_cts
time_design -post_cts -hold
pop_snapshot_stack
create_snapshot -label cts
# ------------------------------------------------------------------------
```

This will result in the following snapshot hierarchy:

```
place {
place_design {
<metrics>
}
}
prects {
opt_design -pre_cts {
<metrics>
}
}
cts {
ccopt_design {
<metrics>
}
time_design -post_cts {
```

```
<metrics>
}
time_design -post_cts -hold {
<metrics>
}
}
```

Related Information

- pop_snapshot_stack
- The Unified Metrics chapter in the *Innovus Stylus Common UI User Guide*

read_metric

```
read_metric
[-help]
[-id snapshotName]
metricsFile
[-merge]
[-previous UniqueID]
```

Loads a previously saved metric file for querying, reporting, and comparing.

The `read_metric` command allows you to manually load previous saved metrics files into an Innovus session. You can load any number of these files. However, these files should have a unique id.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `read_metric` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man read_metric`
`-id snapshotName`	Specifies the name of snapshot to create or overwrite. *Data_type*: string, optional
metricsFile	Specifies the name of the metrics file. *Data_type*: string, required
`-merge`	Merges the metrics into the named snapshot, instead of replacing. *Data_type*: bool, optional
`-previous UniqueID`	Specifies the uuid of previous snapshots for the new pending snapshot. *Data_type*: string, optional

Example

The following code shows the usage of the `read_metric` command along with the `write_metric` command:

```
# ------------------------------------------------------------------

# Generate a XML file for all timing, power, and flow metrics from

# from prects to postcts for a previous run (id v101)

# ------------------------------------------------------------------

read_metric -id v101 run101/dbs/postcts.enc.dat/design.metrics
```

```
write_metric -id v101 -format xml -labels prects-postcts v101.xml
```

Related Information

- write_metric
- The Unified Metrics chapter in the *Innovus Stylus Common UI User Guide*

read_metric_config

```
read_metric_config
[-help]
fileName
[-format {html excel text vivid}]
```

Reads the YAML file written by the `write_metric_config` command.

Parameters

fileName	Specifies the name to read metric configuration from. *Data_type*: string, required
`-format {html excel text vivid}`	Specifies the configuration for specified `report_metric` format. *Data_type*: enum, optional
`-help`	Outputs a brief description that includes type and default information for each `read_metric_config` parameter. For a detailed description of the command and all of its parameters, use the man command: `man read_metric_config`

Example

The following example writes the YAML structure in the `text` format in `small.text`, reads the YAML file, and generates the report in the text format.

```
> write_metric -format text small.text ;

> read_metric_config -format text small.yaml

> report_metric  -format text
```

Related Information

- report_metric
- `write_metric_config`
- The Unified Metrics chapter in the *Innovus Stylus Common UI User Guide*

read_metric_html_package

```
read_metric_html_package
[-help]
filename
```

Overwrites the HTML for `report_metric -format vivid` and the vivid page configuration (accessible via `read_metric_config`/`write_metric_config`) with the contents from the specified *`filename`*.

Parameters

`filename`	Specifies the name of the file from where the tool will read the metric package. If the specified *`filename`* is not a supported format, the tool will give an error message that the file format is invalid. *Data_type*: string, required
`-help`	Outputs a brief description that includes the type and default information for each `read_metric_html_package` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man read_metric_html_package`

Related Information

- The Unified Metrics chapter in the *Innovus Stylus Common UI User Guide*
- `read_metric_config`
- `report_metric`
- `write_metric_config`

report_metric

```
report_metric
[-help]
pattern
-format {html json excel text vivid}
[-id metricID]
[-names listOfNames]
[-out_file fileName]
```

Allows the user to generate reports in various file formats.

Parameters

`-format {html json excel text vivid}`	Specifies the output type. *Data_type*: enum, optional
`-help`	Outputs a brief description that includes type and default information for each `report_metric` parameter. For a detailed description of the command and all of its parameters, use the man command: `man report_metric`
`-id metricID`	Specifies the ID of metrics to report. Supports multiple metric IDs for excel and html formats. The first ID specified will be considered the "gold" value for comparison with other IDs. *Data_type*: string, optional
`-names listOfNames`	Specifies the list of names. *Data_type*: string, optional **Note**: Wildcards are supported.
pattern	Specifies the pattern to match. *Data_type*: string, optional
`-out_file fileName`	Specifies the name of the target file. *Data_type*: string, optional

Example

The following example writes the YAML structure in the `text` format in `small.text`, reads the YAML file, and generates the report in the text format.

```
> write_metric_config -format text small.text ;

> read_metric_config -format text small.yaml

> report_metric  -format text
```

Related Information

- read_metric_config
- `write_metric_config`
- The Unified Metrics chapter in the *Innovus Stylus Common UI User Guide*

report_qor

```
report_qor
[-help]
[pattern]
[-format {html json}]
[-id metricID]
[-names listOfNames]
[-out_file fileName]
```

Allows the user to generate reports in various file formats.

Parameters

`-help`	Prints a brief description that includes the type and default information for each system root attribute.
pattern	Specifies the pattern to be matched. *Data_type*: string, optional
`-out_file` *fileName*	Specifies the name of the target file. *Data_type*: string, optional
-format {html json}	Specifies the output format. *Data_type*: enum, optional
`-id` *metricIDs*	Specifies the IDs of metrics to report. Multiple IDs will produce a comparison, first is gold. *Data_type*: string, optional
`-names` *listOfNames*	Specifies the list of names. You can use wildcards. *Data_type*: string, optional

Related Information

- The Unified Metrics chapter in the *Innovus Stylus Common UI User Guide*

reset_metric_config

```
reset_metric_config
```

Removes all metric configuration.

Parameters

None

Related Information

- The Unified Metrics chapter in the Innovus Stylus Common UI User Guide

reset_metric_run_id

`reset_metric_run_id`

Clears the run ID and the related data from the current state so that `flowtool -ref_run` or similar command does not add snapshots to the original reference run in AUM.

Parameters

None

Related Information

- `flowtool`
- The Unified Metrics chapter in the *Innovus Stylus Common UI User Guide*

run_metric_category

```
run_metric_category
[-help]
-category category_name
```

Runs the specified category.

Parameters

`-help`	Describes the command usage.
`-category category_name`	Specifies the name of the category, such as design, setup, hold, clock, power, route, check, flow, to be run. Each category updates the metric associated with that category. *Data_types*: string, optional

Related Information

- The Unified Metrics chapter in the *Innovus Stylus Common UI User Guide*

set_metric

```
set_metric
-name metric_name
-help
-value metric_value
```

Sets any defined metric to a value.

Parameters

`-name` *`metric_name`*	Specifies the name of the metric. *Data_type*: string, required
`-help`	Outputs a brief description that includes type and default information for each `set_metric` parameter. For a detailed description of the command and all of its parameters, use the man command: `man set_metric`
`-value` *`metric_value`*	Specifies the value of the metric. *Data_type*: string, required

Examples

The following command sets the value `.base_cell.name == HDR*` to the metric `design.switch_cells`.

```
innovus 1> set_metric -name design.switch_cells [get_db insts -if {.base_cell.name == HDR*}]
```

Related Information

- The Unified Metrics chapter in the Innovus Stylus Common UI User Guide

set_metric_alias

```
set_metric_alias
[-help]
-from old_name
[-no_overwrite]
[-order order_of_priority]
-to new_name
```

Renames a metric name, prefixed with the name of a layer, analysis view, path group etc, to another name. For example, you can rename `path_group:reg2reg` to `reg2reg` using the `set_metric_alias` command.

Parameters

`-from` *`old_name`*	Specifies the metric object name to be changed. *Data_type*: string, required
`-help`	Outputs a brief description that includes type and default information for each `set_metric_alias` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man set_metric_alias`.
`-no_overwrite`	Specifies to not overwrite an already set alias. *Data_type*: bool, optional
`-order` *`order_of_priority`*	Specifies the order of priority of metric object in tables. *Data_type*: int, optional
`-to` *`new_name`*	Specifies the new name for the metric object. *Data_type*: string, required

Example

- The following command renames `path_group:reg2reg` to `reg2reg` with priority order `1`.

  ```
  > set_metric_alias -from path_group:reg2reg -to reg2reg -order 1
  ```

- The following command renames `path_group:reg2cgate` to `reg2cgate` with priority order `2`.

  ```
  > set_metric_alias -from path_group:reg2cgate -to reg2cgate -order 2
  ```

- The following command queries the set of aliases:

  ```
  > get_metric_alias
  ```

  Output

  ```
  >path_group:reg2reg {to reg2reg order 1} path_group:reg2cgate {to reg2cgate order 2}
  ```

Related Information

- The Unified Metrics chapter in the *Innovus Stylus Common UI User Guide*

set_metric_header

```
set_metric_header
[-id db_id]
-name header_name
-help
-value header_value
```

Sets the name of a metric header.

Parameters

`-id db_id`	Specifies the database ID. *Data_type*: string, optional
`-name header_name`	Specifies the name of the header. *Data_type*: string, required
`-help`	Outputs a brief description that includes type and default information for each `set_metric_header` parameter. For a detailed description of the command and all of its parameters, use the man command: `man set_metric_header`
`-value header_value`	Specifies the value of the header. *Data_type*: string, required

Related Information

- The Unified Metrics chapter in the *Innovus Stylus Common UI User Guide*
- `delete_metric_header`

update_metric

```
update_metric
[-help]
[-format {html excel text}]
[-graph_type {treemap line none}]
[-group group_title]
[-header header_name]
[-link_file_metric metric_name]
[-link_table table]
[-metric metric_name]
[-no_shared_max]
[-renderer html_format]
[-show_graph]
-table table_id
[-title html_table_title]
```

Updates a metric.

Parameters

`-format {html excel text}`	Specifies configuration for the specified `report_metric` format. *Data_type*: enum, optional
`-graph_type {treemap line none}`	Specifies the type of graph to use in the table (treemap |line |none). *Default*: `line` *Data_type*: enum, optional
`-group group_title`	Specifies the group title to be used in the html table. *Data_type*: string, optional
`-header header_name`	Specifies the header name to be added. *Data_type*: string, optional
`-help`	Prints out the command usage. For a detailed description of the command, use the `man` command: `man update_metric`
`-metric metric_name`	Specifies the metric name to be added. *Data_type*: string, optional
`-no_shared_max`	When specified for a `-renderer` histogram metric, the Y axis of all the histograms in a single column stop sharing the same max Y value. *Data_type*: bool, optional

`-link_file_metric` *metric_name*	Specifies the name of the metric containing filename to open when the metric value is clicked. *Data_type*: string, optional
`-link_table` *table*	Specifies the table to scroll to when the metric value is clicked. *Data_type*: string, optional
`-renderer` *html_format*	Forces a specific html format renderer for the histogram metric. *Data_type*: string, optional
`-show_graph`	Specifies whether to show this metric in a comparison graph. *Data_type*: bool, optional
`-table` *table_id*	Specifies the id of table to which the metric will be added. *Data_type*: string, required
`-title` *html_table_title*	Specifies the title to be used in the html table. *Data_type*: string, optional

Example

The following sequence of commands create the `table_s` metric table and update the table using the `update_metric` command.

```
innovus 1> create_metric_page -name User

innovus 2> create_metric_table_heading -id statistics -page User  -title "Average Timing Values Per Snapshot"

innovus 3> create_metric_table -id table_s -heading statistics -per_snapshot -type  horizontal

innovus 4> update_metric -table table_s -metric user.avg_slack -group "Timing Statistics" -title avg_slack

innovus 5> update_metric -table table_s -metric user.arith_mean -group "Timing Statistics" -title arith_mean

innovus 6> update_metric -table table_s -metric user.geom_mean -group "Timing Statistics" -title geom_mean

innovus 7> update_metric -table table_s -metric user.rms -group "Timing Statistics" -title rms
```

Related Information

- The Unified Metrics chapter in the *Innovus Stylus Common UI User Guide*

update_metric_definition

```
update_metric_definition
[-help]
[-compare compareFunction]
[-description metricDescription]
[-inheritance_tcl TclName]
-name metric_name
[-no_inheritance]
[-precision number_of_decimal_places]
[-target metric_name]
[-tcl {TCL_command_list}[-tools listOfTools]]
[{-id databaseID}
[-threshold reporting_unit]
[-type metricType]
[-units reporting_unit]
[-warning threshold]
```

Updates the specified properties of an already-existing metric definition with the same name.

Note: If a metric does not exist with the name, the command returns an error.

Parameters

`-compare compareFunction`	Specifies the compare function, `[moreBetter|lessBetter|moreBetterBelowTarget|lessBetterAboveTarget]`. The comparison options show the good or bad color when the value is below or above the target value as appropriate. *Data_type*: string, optional
`-description metricDescription`	Specifies the description of the metric. *Data_type*: string, optional
`-id databaseID`	Specifies the database ID. *Data_type*: string, optional
`-inheritance_tcl TclName`	Specifies Tcl for combining children values. The metric is not executed in the global namespace. The equivalent proc args is {children}. *Data_type*: string, optional
`-name metric_name`	Specifies the name of the metric. *Data_type*: string, required
`-no_inheritance`	Disables the inheritance of the metric. *Data_type*: bool, optional

`-help`	Outputs a brief description that includes type and default information for each `update_metric_definition` parameter. For a detailed description of the command and all of its parameters, use the man command: `man update_metric_definition`
`-precision` *`number_of_decimal_places`*	Specifies the number of decimal places. *Data_type*: int, optional *Default*: 0
`-target` *`metric_name`*	Specifies the target or the name of metric containing the target value. *Data_type*: string, optional
`-tcl { ` *`TCL_command_list`* ` }`	Specifies the list of TCL commands or proc to generate (return) the metric which is of the format `{<value> <unit>}`. *Data_type*: string, optional
`-threshold` *`reporting_unit`*	Specifies the threshold for comparing numbers in HTML. If the delta between the two numbers being compared is less than the threshold then the metric will be colored “yellow”. If it is outside the threshold, then it is colored red or green depending on the compare function i.e. whether more is better or less is better. *Data_type*: string, optional
`-tools` *`listOfTools`*	Specifies the list of tools in which the tcl code will run. *Data_type*: string, optional

`-type` *`metricType`*	Specifies the type of metric.
	Data_type: string, optional
	The various metrics types are:
	• imageplot: A metric type that the renders to an image in the HTML report.
	• Format:
	`<name1> { \`
	` fill_color "<color>" \`
	`  stroke_color "<color>"\`
	`  data <list of data sets>`
	`} \`
	`<name2> { \`
	`<sub-name1> { \`
	`  fill_color <color> \`
	`  data <list of data sets> \`

```
  } \
  <sub-name2> {...} \
}
```

Where a data set can be one or more of the following:

The data on each layer is a string of coordinates (separated by sequences of whitespace and comma) with types. These types are:

- R - rectangle (x, y, width, height)
- B - bounding box (x_min, y_min, x_max, y_max)
- G - closed polygon (x, y, x, y, ...)
- L - open polyline (x, y, x, y, ...)
- C - circle (x, y, r)
- P - point (x, y)

For rectangles, boxes, circles and points, multiple shapes can be defined without having to repeat the type letter. e.g. R1,2,3,4,5,6,7,8 is two rectangles.

A value can be defined for shapes by adding a V suffix to the type; this value can then be used to color the shapes according to the value for heat maps. The value is always the first number after the type. This means that two points with values 20 and 40 may look like: PV20,1,2 40,6,5 or: PV20,1,2PV40,6,5.

Example imageplot TCL dict data:

```
macros { \
fill_color "#FF0000" \
  stroke_color "#0000FF88" \
  data "R5,8,1,2 6,9,2,1 G1,2,3,4,5,6,7,8" \
} \
drcs { \
placement { \
    fill_color "#FF0000" \
    data "P3,4" \
  } \
  routing {...} \
}
```

Layers in the image are named (macros, congestion, normal, other in the above example) and support hierarchy. Layers should not be nested beyond one level deep. The hierarchy is used in the renderer (see below) to toggle the visibility of the layers. It is an error for any layer to be called data (reserved to keep hierarchy clear).

Colors can be defined either as hex or RGB values with optional opacity. Hex values are prefixed with #. RGB values are separated with whitespace.

- #FF0000: Red
- #00FF0040: Green (25% opaque)
- 0 0 255: Blue
- 255 0 255 192: Magenta (75% opaque)

- histogram: The data will be structured as a set of series with shared X-axis labels that are rendered as a stacked histogram.

  Format (TCL dict):

```
labels <list of labels> \
axis_x <name> \
axis_y <name> \
series { \
<series1> { \
   values <list of buckets \
   color "#FF0000" \
} \
<series2> { \
   values {2 1 0 0} \
   color "#0000FF" \
} \
}
```

  Example:

```
get_metric timing.setup.histogram
timing.setup.histogram {labels {-0.320 -0.240 -0.160 -0.080} \
                        series { \
                          reg2reg {color #FF0000 values {43 3 2
3 0}} \
                          reg2cgate {color #0000FF values {22 4
5 0 0}} \
                        } \
                        axis_x WNS(ns) \
                        axis_y {Failing End Points} \
                        }
```

  The labels (a list of strings, shown in the order provided) define the X-

	axis labels for the histogram, the values in each series define the Y-axis values.
	Colors can be defined either as hex or RGB values with optional opacity. Hex values are prefixed with #. RGB values are separated with whitespace.
	`#FF0000 - Red`
	`0 0 255 - Blue`
	• html: Supports the inclusion of raw HTML in a metric. You can place simple html into tables allowing simple links to reports. For example, the following command defines `test_html` as an html type of metric:
	`define_metric -name test_html -type html`
	You can the html metric, test_html, using the `set_metric` command:
	`set_metric -name test_html -value {<a href="url">test</a>}`

`-units` *reporting_unit*	Specifies the unit for reporting. *Data_type*: string, optional
`-warning` *threshold*	Specifies the warning threshold or name of metric containing the threshold value. *Data_type*: string, optional

Example

- The following command updates the `-precision` attribute of the `timing.setup.wns` metric:

```
innovus 1> update_metric_definition -name timing.setup.wns -precision 2
innovus 2> get_metric_definition -name timing.setup.wns
metric timing.setup.wns units ns
type slack description {Worst negative
setup slack (all path groups)}
dp 2
cmp moreBetterBelowTarget
target 0
```

Related Information

- The Unified Metrics chapter in the Innovus Stylus Common UI User Guide

write_metric

```
write_metric
[pattern]
[-format output_format]
[-id metric_id]
[-merge]
[-names list_of_patterns]
[-out_file fileName]
```

Writes the current set of metrics to a file or stdout.

The write_metric command is used by `write_db` to write the metrics in the JSON format in the database directory. Super commands use `write_metric` to write metrics into the log file (stdout). This can also be used to write metrics in various formats by the user.

Parameters

`-format output_format`	Specifies the output format, json, xml, csv, tcl. *Default*: JSON *Data_type*: string, optional
`-help`	Outputs a brief description that includes type and default information for each `write_metric` parameter. For a detailed description of the command and all of its parameters, use the man command: `man write_metric`
`-id metric_id`	Specifies a list of labels from read_metrics. *Default*: Current metric *Data_type*: string, optional
`-merge`	Merges the results into an existing file. *Data_type*: bool, optional
`-names list_of_patterns`	Specifies the list of patterns to match snapshot names. *Data_type*: string, optional
`pattern`	List of regular expressions to chose metrics to write. Default is all metrics (*). *Data_type*: string, optional
`-out_file fileName`	Specifies the name of the file in which metrics is written. *Data_type*: string, optional

Example

The following code shows the usage of the `write_metric` command:

```
# ------------------------------------------------------------------
# Generate a XML file for all timing, power, and flow metrics from
# from prects to postcts for a previous run (id v101)
# ------------------------------------------------------------------
read_metric -id v101 run101/dbs/postcts.enc.dat/design.metrics
write_metric -id v101 -format xml
```

Related Information

- `read_metric`
- `write_db`
- The Unified Metrics chapter in the *Innovus Stylus Common UI User Guide*.

write_metric_config

```
write_metric_config
[-help]
fileName
[-format {html excel text vivid}]
```

Writes the YAML structure for the requested format (text, html, or excel) to a file. As the configuration is written in YAML format, you can customize the page layout and table content you need in the report such as adding pages and tables, or adding new metrics to existing tables, removing metrics from existing tables, and rearranging the metric ordering.

Parameters

fileName	Specifies the name of the file in which metric config is written. *Data_type*: string, required
-format {html excel text vivid}	Specifies the config for specified report_metric format. *Data_type*: enum, optional
-help	Outputs a brief description that includes type and default information for each write_metric_config parameter. For a detailed description of the command and all of its parameters, use the man command: man write_metric_config

Example

generates the report in the text format.

> write_metric_config -format text small.text ; *# Edit the summary table*

> read_metric_config -format text small.yaml

> report_metric -format text

Related Information

- The Unified Metrics chapter in the *Innovus Stylus Common UI User Guide*
- read_metric_config
- report_metric

rename_snapshot

```
rename_snapshot
[-help]
[-branch newBranch]
[-id metricID]
[-name newName]
-uuid snapshotuuid
```

Allows you to rename existing snapshots.

`rename_snapshot` does not support canonical paths and cannot change the location of a snapshot within the hierarchy.

Parameters

`-branch` *`newBranch`*	Specifies the new branch for the snapshot. *Data_type*: string, optional
`-help`	Outputs a brief description that includes type and default information for each `rename_snapshot` parameter. For a detailed description of the command and all of its parameters, use the man command: `man rename_snapshot`
`-id` *`metricID`*	Specifies the metric database ID to change. If you do not specify any id, the tool will take current database as default. *Data_type*: string, optional **Note**: You can find the uuid of a snapshot by using `get_metric` for the uuid metric.
`-name` *`newName`*	Specifies the new name for the snapshot. The tool will error out if you set the name to empty string. *Data_type*: string, optional
`-uuid` *`snapshotuuid`*	Specifies the snapshot UUID. The tool will error out if there is no snapshot with the provided uuid. *Data_type*: string, required

Related Information

- `get_metric`

delete_metric_header

```
delete_metric_header
[-help]
[-id database_id]
-name header_name
```

Deletes the metric header.

Parameters

`-help`	Prints out the command usage. For a detailed description of the command, use the man command: `man delete_metric_header`
`-id` *`database_id`*	Specifies the database ID *Data_type*: string, optional
`-name` *`header_name`*	Specifies the header name to be deleted. *Data_type*: string, required

Example

- The following command deletes the metric header, `User`.

  ```
  delete_metric_header -name User
  ```

Related Information

- `set_metric_header`
- The Unified Metrics chapter in the Innovus Stylus Common UI User Guide

Appendix A: Database Object Information

Please find the document at the following link: Stylus Common UI Database Object Information

Index

A

add_area_io_fillers
add_area_io_rows
add_clock_tree_exclusion_drivers
add_clock_tree_source_group_roots
add_connection_to_neighbor_partition
add_decap_cell_candidates
add_decaps
add_decomp_filler
add_dummy_boundary_wires
add_endcaps
add_endcaps_avoid_two_sites_cell_abut
add_endcaps_bottom_edge
add_endcaps_boundary_tap
add_endcaps_boundary_tap_swap_flow
add_endcaps_cells
add_endcaps_create_rows
add_endcaps_double_height_incorner_protrusion
add_endcaps_flip_y
add_endcaps_incremental_left_edge
add_endcaps_incremental_right_edge
add_endcaps_left_bottom_corner
add_endcaps_left_bottom_corner_even
add_endcaps_left_bottom_corner_neighbor
add_endcaps_left_bottom_corner_odd
add_endcaps_left_bottom_edge
add_endcaps_left_bottom_edge_even
add_endcaps_left_bottom_edge_neighbor
add_endcaps_left_bottom_edge_odd
add_endcaps_left_corner_bottom_border
add_endcaps_left_corner_top_border
add_endcaps_left_edge

add_endcaps_left_edge_bottom_border
add_endcaps_left_edge_even
add_endcaps_left_edge_odd
add_endcaps_left_edge_top_border
add_endcaps_left_top_corner
add_endcaps_left_top_corner_neighbor
add_endcaps_left_top_corner_odd
add_endcaps_left_top_edge
add_endcaps_left_top_edge_even
add_endcaps_left_top_edge_neighbor
add_endcaps_left_top_edge_odd
add_endcaps_min_horizontal_channel_width
add_endcaps_min_jog_height
add_endcaps_min_jog_width
add_endcaps_min_vertical_channel_width
add_endcaps_prefix
add_endcaps_right_bottom_corner
add_endcaps_right_bottom_corner_even
add_endcaps_right_bottom_corner_neighbor
add_endcaps_right_bottom_corner_odd
add_endcaps_right_bottom_edge
add_endcaps_right_bottom_edge_even
add_endcaps_right_bottom_edge_neighbor
add_endcaps_right_bottom_edge_odd
add_endcaps_right_corner_bottom_border
add_endcaps_right_corner_top_border
add_endcaps_right_edge
add_endcaps_right_edge_bottom_border
add_endcaps_right_edge_even
add_endcaps_right_edge_odd
add_endcaps_right_edge_top_border
add_endcaps_right_top_corner
add_endcaps_right_top_corner_even
add_endcaps_right_top_corner_neighbor
add_endcaps_right_top_corner_odd
add_endcaps_right_top_edge
add_endcaps_right_top_edge_even
add_endcaps_right_top_edge_neighbor
add_endcaps_right_top_edge_odd
add_endcaps_top_bottom_edge
add_endcaps_top_edge
add_endcaps_use_even_odd_sites

add_endcaps_wall_keepout_from_vertical_boundary
add_endcaps_wall_offset
add_endcaps_wall_pitch
add_endcaps_wall_to_convex_corner_spacing
add_fences
add_filler_gaps
add_fillers
add_fillers_avoid_abutment_patterns
add_fillers_cell_name_style
add_fillers_cells
add_fillers_check_different_cells
add_fillers_check_drc
add_fillers_create_rows
add_fillers_distribute_implant_evenly
add_fillers_eco_mode
add_fillers_honor_preroute_as_obs
add_fillers_horizontal_exception_cell
add_fillers_horizontal_max_length
add_fillers_horizontal_repair_cell
add_fillers_keep_fixed
add_fillers_no_single_site_gap
add_fillers_prefix
add_fillers_preserve_user_order
add_fillers_scheme
add_fillers_swap_cell
add_fillers_vertical_stack_exception_cell
add_fillers_vertical_stack_left_edge_exception_cell
add_fillers_vertical_stack_max_length
add_fillers_vertical_stack_repair_cell
add_fillers_vertical_stack_repair_edge
add_fillers_vertical_stack_right_edge_exception_cell
add_fillers_with_drc
add_fillers_y_flip_type
add_gate_array_filler
add_interposer_route_block
add_io_buffers
add_io_fillers
add_io_insts
add_io_row_fillers
add_line_end_track
add_metal_fill
add_metal_fill_signoff

add_mim_cap
add_notch_fill
add_obj_on_bump
add_partition_feedthrus
add_pg_fill
add_physical_cell
add_power_mesh_colors
add_power_switches
add_reinforce_pg_auto_ir_fix_effort
add_reinforce_pg_critical_path_slack
add_reinforce_pg_irdrop_hierarchical_block
add_reinforce_pg_irdrop_hierarchical_top_rail_analysis_directory
add_reinforce_pg_respect_defined_nets
add_reinforce_pg_respect_routes
add_reinforce_pg_respect_stdcell_geometry
add_reinforce_pg_timing_aware_effort
add_repeater_by_rule
add_rings
add_rings_avoid_short
add_rings_break_core_ring_io_list
add_rings_continue_on_no_selection
add_rings_detailed_log
add_rings_extend_block_ring_search_distance
add_rings_extend_core_ring_search_distance
add_rings_extend_merge_with_pre_wires
add_rings_extend_over_row
add_rings_extend_search_nets
add_rings_extend_stripe_search_distance
add_rings_gap_width_without_io
add_rings_ignore_rows
add_rings_max_via_size
add_rings_orthogonal_only
add_rings_skip_shared_inner_ring
add_rings_skip_via_on_pin
add_rings_skip_via_on_wire_shape
add_rings_spacing_from_block
add_rings_split_long_via
add_rings_stacked_via_bottom_layer
add_rings_stacked_via_top_layer
add_rings_target
add_rings_via_using_exact_crossover_size
add_rings_wire_center_offset

add_route_via_defs
add_route_vias_auto
add_route_vias_ndr_only
add_sai_boundary_flops
add_sdp_rows_blockages
add_spare_insts
add_split_power_vias
add_stripes_allow_jog
add_stripes_allow_non_preferred_dir
add_stripes_area
add_stripes_blocks_without_same_net
add_stripes_break_at
add_stripes_continue_on_no_selection
add_stripes_detailed_log
add_stripes_domain_offset_from_core
add_stripes_extend_to_closest_target
add_stripes_extend_to_first_ring
add_stripes_ignore_block_check
add_stripes_ignore_block_ring_during_break
add_stripes_ignore_drc
add_stripes_ignore_non_default_domains
add_stripes_in_cell_allow_shift
add_stripes_in_cell_only
add_stripes_keep_pitch_after_snap
add_stripes_mask_color_balance
add_stripes_max_extension_distance
add_stripes_max_via_size
add_stripes_merge_with_all_layers
add_stripes_mesh_via
add_stripes_offset_from_core
add_stripes_opt_stripe_for_routing_track
add_stripes_orthogonal_offset
add_stripes_orthogonal_only
add_stripes_over_row_extension
add_stripes_partial_set_through_domain
add_stripes_remove_floating_stapling
add_stripes_remove_floating_stripe_over_block
add_stripes_remove_stripe_under_ring
add_stripes_respect_routes
add_stripes_route_over_rows_only
add_stripes_rows_without_stripes_only
add_stripes_same_size_stack_vias

add_stripes_skip_via_on_pin
add_stripes_skip_via_on_wire_shape
add_stripes_spacing_from_block
add_stripes_spacing_type
add_stripes_split_long_via
add_stripes_split_vias
add_stripes_split_wire_spacing
add_stripes_split_wire_weight
add_stripes_split_wire_width
add_stripes_stacked_via_bottom_layer
add_stripes_stacked_via_top_layer
add_stripes_stapling_extend_to_minimum_spacing
add_stripes_stapling_nets_style
add_stripes_stapling_shift
add_stripes_stop_at_closest_target
add_stripes_stop_at_last_wire_for_area
add_stripes_stripe_min_length
add_stripes_stripe_min_width
add_stripes_switch_cell_name
add_stripes_switch_layer_overlap_length
add_stripes_trim_antenna_back_to_shape
add_stripes_trim_antenna_max_distance
add_stripes_trim_stripe
add_stripes_use_exact_spacing
add_stripes_use_point_to_point_route
add_stripes_use_stripe_width
add_stripes_via_using_exact_crossover_size
add_stripes_width_file
add_tap_walls
add_target_pg_allow_weak_connect
add_target_pg_max_extension_distance
add_target_pg_pins
add_target_pg_pins_group_distance
add_target_pg_respect_routes
add_target_pg_share_resource
add_tieoffs
add_tieoffs_cells
add_tieoffs_create_hports
add_tieoffs_honor_dont_touch
add_tieoffs_honor_dont_use
add_tieoffs_max_distance
add_tieoffs_max_fanout

add_tieoffs_module_prevention
add_tieoffs_prefix
add_tieoffs_report_hports
add_to_collection
add_track_fill
add_tracks
add_tsv
add_via_fill
add_well_taps
add_well_taps_avoid_abutment
add_well_taps_avoid_vertical_well_abutment
add_well_taps_block_boundary_only
add_well_taps_bottom_tap_cell
add_well_taps_bottom_termination_cell
add_well_taps_cell
add_well_taps_channel_offset
add_well_taps_check_channel
add_well_taps_column_cells
add_well_taps_create_rows
add_well_taps_disable_check_zone_at_boundary
add_well_taps_insert_cells
add_well_taps_rule
add_well_taps_site_offset
add_well_taps_tap_function_cells
add_well_taps_tap_termination_alignment
add_well_taps_termination_align
add_well_taps_termination_cells
add_well_taps_top_tap_cell
add_well_taps_top_termination_cell
add_well_taps_vertical_boundary_spacing
add_well_taps_well_cut_cells
add_what_if_shapes
alias
align_partition_clones
align_pins
align_selected
all_fanin
all_fanout
all_inputs
all_instances
all_outputs
all_registers

analyze_ir_paths
analyze_paths_by_bottleneck
analyze_paths_by_clock_domain
analyze_paths_by_critical_false_path
analyze_paths_by_default_path_group
analyze_paths_by_drv
analyze_paths_by_hier_port
analyze_paths_by_hierarchy
analyze_paths_by_view
analyze_resistance
analyze_resistance
analyze_resistance
analyze_resistance
append_to_collection
apply_safety_mechanism
assemble_design
assemble_proto_model
assign_bumps
assign_clock_tree_source_groups
assign_io_pins
assign_partition_pins
assign_pg_bumps
assign_pins_advanced_node_rule_support
assign_pins_allow_non_ndr_net_pins_on_ndr_tracks
assign_pins_allow_unconnected_in_abutted_edge
assign_pins_blocked_boundary_macro_distance
assign_pins_edit_in_batch
assign_pins_force_abutment_with_fixed
assign_pins_max_channel_width_as_abutted
assign_pins_max_distance_pair
assign_pins_off_stripe
assign_pins_pin_to_stripe_distance
assign_pins_pin_to_via_distance_non_preferred_direction
assign_pins_pin_to_via_distance_preferred_direction
assign_pins_promoted_macro_bottom_layer
assign_pins_promoted_macro_top_layer
assign_pins_restricted_boundary_macro_distance
assign_pins_strict_abutment
assign_signal_to_bump
assign_tsv
auto_file_dir
auto_file_prefix

B

bit_blasted_port_style
bitblast_ports
budget_abutted
budget_accumulated
budget_boundary_condition_template_path
budget_buffer_delay_adjustment
budget_buffer_delay_lib_cell
budget_buffer_delay_selection_effort
budget_constant_model
budget_distribute_mmmc
budget_drive_cell
budget_fix_top_level_paths
budget_handle_complex_sdc
budget_honor_report_timing_format
budget_ignore_dont_touch
budget_include_latency
budget_include_wire_loads_in_lib
budget_input_load
budget_input_transition
budget_justify
budget_keep_pin_list_for_block_ports
budget_latency_on_clocks
budget_local_latency
budget_local_uncertainty
budget_make_negative_input_delay_zero
budget_master_clone
budget_no_false_paths_for_unconstrained_ports
budget_no_hold_view
budget_no_setup_view
budget_override_net_cap
budget_pin_load
budget_report_dir
budget_report_negative_slack_on_ports
budget_report_or_update_budget
budget_snap_feedthru_budget_to
budget_snap_input_budget_to
budget_snap_negative_only
budget_snap_output_budget_to
budget_stage_based_fanout_driving_factor
budget_stage_based_partition_multiplier
budget_stage_based_weight

budget_top_level
budget_top_level_delay_per_length
budget_top_level_min_delay_per_net
budget_use_boundary_condition
budget_use_real_cell_for_timing_model
budget_virtual_opt_engine
budget_write_constraints_for_clock_output_ports
budget_write_false_path_for_hold
budget_write_latency_per_clock

C

ccopt_auto_limit_insertion_delay_factor
ccopt_design
ccopt_merge_clock_gates
ccopt_merge_clock_logic
change_current_design
checkDesign
check_ac_limit_additional_individual_violation_report
check_ac_limit_avg_recovery
check_ac_limit_current_file
check_ac_limit_current_scale_factor
check_ac_limit_current_scale_table
check_ac_limit_default_freq_for_unconstrained_nets
check_ac_limit_delta_temperature
check_ac_limit_delta_temperature_layer_list
check_ac_limit_detailed
check_ac_limit_effort_level
check_ac_limit_em_cdf_percentage
check_ac_limit_em_limit_scale_factor
check_ac_limit_em_limit_scale_table
check_ac_limit_em_res_width
check_ac_limit_em_temperature
check_ac_limit_em_temperature_layer_list
check_ac_limit_em_threshold
check_ac_limit_enable_seb
check_ac_limit_env_temperature
check_ac_limit_extraction_tech_file
check_ac_limit_force_hold_view
check_ac_limit_handle_pin_obs_via
check_ac_limit_ict_em_models
check_ac_limit_ircx_models
check_ac_limit_lifetime

check_ac_limit_max_error
check_ac_limit_method
check_ac_limit_min_peak_duty_ratio
check_ac_limit_min_peak_freq
check_ac_limit_net_file
check_ac_limit_nets
check_ac_limit_out_file
check_ac_limit_peak_td_method
check_ac_limit_seb_lifetime
check_ac_limit_seb_table
check_ac_limit_seb_temperature
check_ac_limit_selected
check_ac_limit_simulation_net
check_ac_limit_simulation_net_file
check_ac_limit_skip_category_mode
check_ac_limit_skip_net
check_ac_limit_skip_net_file
check_ac_limit_toggle
check_ac_limit_use_db_freq
check_ac_limit_use_qrc_tech
check_ac_limit_use_rms_delta_t
check_ac_limit_view
check_ac_limits
check_ac_limits
check_antenna
check_bond_pad_spacing
check_bumps
check_bus_guides
check_clock_tree_convergence
check_connectivity
check_cts_config
check_cut_density
check_design_across_hierarchy
check_design_report_async_pins
check_design_report_scan_pins
check_design_threshold_fanout
check_drc
check_drc_area
check_drc_check_only
check_drc_check_reverse
check_drc_check_routing_halo
check_drc_check_routing_halo_corner

check_drc_disable_rules
check_drc_enable_post_passive_fill
check_drc_exclude_pg_net
check_drc_ignore_cell_blockage
check_drc_ignore_non_rectangle_shapes
check_drc_ignore_trial_route
check_drc_inside_via_def
check_drc_layer_range
check_drc_limit
check_drc_max_wrong_way_halo
check_drc_ndr_spacing
check_drc_report
check_drc_short_only
check_drc_trim_length
check_drc_uncolored
check_drc_use_min_spacing_on_block_obs
check_endcaps
check_filler
check_flip_chip_constraints
check_floorplan
check_floorplan_space
check_flow
check_footprint
check_hier_route
check_io_inst_overlap
check_io_to_bump_connectivity
check_isolated_cuts
check_library
check_macro_lower_left_on_track
check_metal_density
check_module_model
check_multi_cpu_usage
check_netlist
check_noise
check_partition_sdc
check_pg_library
check_pg_shorts
check_pgv
check_pin_assignment
check_place
check_power_domains
check_power_intent

check_power_switches
check_power_vias
check_process_antenna
check_safety_mechanism
check_sdps
check_stacked_die
check_syntax
check_tieoffs
check_timing
check_timing_library
check_tquantus_model_file
check_tracks
check_ufc
check_unique
check_well_antenna
check_well_taps
check_wire_gaps
clearDrc
clearDrc
clear_active_logic_view
clear_all_rulers
clock_design
clock_opt_design
clock_post_route_repair
clock_tree_source_groups
clock_trees
clone_msv_gate
clone_partition_obj
cmd_file
colorize_cell_library
commit_clock_tree_route_attributes
commit_module_model
commit_partitions
commit_power_intent
commit_pushdown_eco
compare_def
compare_model_timing
compare_oa_cellview
compare_pin_assignment_statistics
compare_power_intent
compare_release
compress_special

connect
connect_clock_tree_mesh_drivers
connect_die_package
connect_global_net
connect_hpin
connect_macro_feedthru
connect_pin
convert_blackbox_to_fence
convert_dbu_to_um
convert_fence_to_black_box
convert_fence_to_lef
convert_lib_clock_tree_latencies
convert_proto_model
convert_special_routes_for_eco
convert_stream_to_def
convert_um_to_dbu
copy_bumps
copy_obj
createObstruct
createObstruct
create_active_logic_view
create_analysis_view
create_basic_path_groups
create_blackbox
create_boundary_constraint
create_bump
create_bump_connect_target_constraint
create_bump_grid
create_bus_guide
create_channel_density_control
create_clock
create_clock_tree
create_clock_tree_source_group
create_clock_tree_spec
create_constraint_mode
create_delay_corner
create_diode
create_esubpart
create_exclusive_groups
create_failure_mode
create_fillet
create_flexfiller_route_blockage

create_flexible_htree
create_floorplan
create_flow
create_flow_step
create_gate_array_filler_group
create_generated_clock
create_generated_clock_tree
create_group
create_gui_shape
create_gui_text
create_hinst
create_histogram_png
create_hnet
create_hport
create_ilm_from_files
create_inst
create_inst_space_group
create_interposer_route_block
create_io_row
create_library_set
create_logic_hierarchy
create_marker
create_metric_page
create_metric_table
create_metric_table_heading
create_module
create_net
create_net_group
create_oa_lib
create_oa_lib_property
create_opcond
create_parasitic_coordinate_transform
create_partition
create_partition_feedthru
create_partition_wrapper
create_path_category
create_pg_keep_out
create_pg_model_for_macro_place
create_pg_pin
create_physical_pin
create_pin_blockage
create_pin_group

create_pin_guide
create_pin_text
create_pipeline_bus_guide
create_pipeline_net_group
create_place_blockage
create_place_halo
create_port
create_power_pads
create_property_alias
create_proto_model
create_proto_net_delay_model
create_pushdown_eco
create_rc_corner
create_resize_blockage
create_route_blockage
create_route_blockage_along_track
create_route_halo
create_route_rule
create_route_type
create_row
create_ruler
create_safety_mechanism
create_scan_chain
create_scan_chain_partition
create_sdp
create_shape
create_signal_pin
create_skew_group
create_snapshot
create_spare_module
create_spice_deck
create_stress_rule
create_text
create_timing_budget
create_timing_condition
create_tquantus_model_file
create_tracks
create_via
create_via_definition
create_virtual_shape_on_pg_wires
cts_add_port_driver_max_distance_from_port
cts_add_port_driver_max_distance_from_port_rows

cts_adjacent_rows_legal
cts_allow_non_std_clock_gate_inputs
cts_balance_mode
cts_blackbox_default_driver_base_pin
cts_blackbox_default_load_base_pin
cts_buffer_cells
cts_buffer_cells_leaf
cts_buffer_cells_top
cts_cell_density
cts_cell_halo_mode
cts_cell_halo_rows
cts_cell_halo_sites
cts_cell_halo_x
cts_cell_halo_y
cts_clock_gate_movement_limit
cts_clock_gating_cells
cts_clock_source_cells
cts_clock_tree_source_group_clock_trees
cts_clock_tree_source_input_max_transition_time
cts_cloning_inst_name_suffix
cts_cloning_inst_name_suffix_source_group_assignment
cts_delay_cells
cts_detailed_cell_warnings
cts_exit_if_no_placeable_area
cts_exit_if_skew_target_over_constrained
cts_exit_if_stage_delay_sigma_target_over_constrained
cts_fix_clock_sinks
cts_flexible_htree_enable_metrics_reporting
cts_flexible_htree_partition_boundary_polarity
cts_flexible_htree_placement_legalization_effort
cts_force_target_max_transition_time
cts_ignore_problematic_skew_as_result_of_dont_touch_nets
cts_inst_name_prefix
cts_inverter_cells
cts_inverter_cells_leaf
cts_inverter_cells_top
cts_load_capacitance_cells
cts_logic_cells
cts_manage_power_intent_violations
cts_max_fanout
cts_max_instance_height
cts_max_source_to_sink_net_length

cts_max_source_to_sink_net_length_leaf
cts_max_source_to_sink_net_length_top
cts_max_source_to_sink_net_length_trunk
cts_merge_clock_gates
cts_merge_clock_logic
cts_mixed_fanout_net_type
cts_move_clock_gates
cts_move_logic
cts_net_name_prefix
cts_override_vias
cts_pin_insertion_delay_histogram_bin_size
cts_post_route_enable
cts_post_route_enable_routing_eco
cts_post_route_repair_drv
cts_post_route_repair_skew_by_buffering
cts_primary_delay_corner
cts_primary_reporting_skew_groups
cts_primary_reporting_skew_groups_log_min_max_sinks
cts_repair_congestion
cts_repair_drv_by_buffering
cts_report_skew_groups_only_with_targets
cts_route_clock_tree_nets
cts_route_type_leaf
cts_route_type_top
cts_route_type_trunk
cts_routing_preferred_layer_effort
cts_size_clock_sources
cts_size_logic
cts_skew_group_report_columns
cts_skew_group_report_histogram_bin_size
cts_spec_config_create_clock_tree_source_groups
cts_spec_config_create_clock_tree_source_groupscts_spec_config_base_pin_trace_through_to
cts_spec_config_create_generator_skew_groups_name_prefix
cts_spec_config_create_reporting_only_skew_groups
cts_spec_config_trace_multi_output_cells_with_single_clock_output
cts_target_max_capacitance
cts_target_max_capacitance_leaf
cts_target_max_capacitance_top
cts_target_max_capacitance_trunk
cts_target_max_stage_delay_sigma
cts_target_max_transition_time
cts_target_max_transition_time_leaf

cts_target_max_transition_time_top
cts_target_max_transition_time_trunk
cts_target_skew
cts_timing_connectivity_based_skew_groups
cts_timing_connectivity_based_skew_groups_balance_master_clocks
cts_timing_connectivity_info
cts_top_fanout_threshold
cts_update_clock_latency
cts_use_inverters
cts_use_receiver_model_capacitance_for_drv
current_instance
cut_box_list_from_power_domain
cut_power_domain_by_overlaps

D

decompress_special
decrypt
define_add_on_module_model
define_attribute
define_bundled_bus
define_feature
define_metric
define_proc
define_proc_arguments
define_property
define_variables
deleteEmptyModule
delete_all_cell_padding
delete_all_density_area
delete_all_floorplan_objs
delete_all_partition_cuts
delete_all_partition_feedthrus
delete_all_power_preroutes
delete_all_scan_cells
delete_all_signal_preroutes
delete_analysis_view
delete_area_io_fillers
delete_assigns
delete_base_cell_edge_type
delete_black_boxes
delete_buffer
delete_buffer_trees

delete_bump_connect_target_constraints
delete_bump_grid
delete_bumps
delete_bus_guides
delete_bus_net
delete_cell_edge_spacing
delete_cell_obs
delete_cell_padding
delete_cell_stack_area
delete_cell_stack_group
delete_cell_virtual_align
delete_clock_tree_drivers
delete_clock_tree_source_group
delete_clock_tree_spec
delete_clock_trees
delete_dangling_ports
delete_decaps
delete_drc_markers
delete_empty_hinsts
delete_exclusive_groups
delete_filler
delete_fillet
delete_flexible_htrees
delete_floating_constants
delete_floating_nets
delete_flow_config
delete_gate_array_filler_group
delete_global_net_connections
delete_hinst
delete_hpin
delete_inst
delete_inst_padding
delete_inst_space_group
delete_integration_route_constraints
delete_io_fillers
delete_io_insts
delete_io_row_fillers
delete_markers
delete_metal_fill
delete_metric
delete_metric_header
delete_metric_page

delete_metric_run
delete_metric_table
delete_metric_table_heading
delete_mim_cap
delete_net_group
delete_nets
delete_nets_from_net_group
delete_notch_fill
delete_obj
delete_partition_cuts
delete_partitions
delete_path_category
delete_pg_keep_out
delete_pg_pins
delete_pin_blockages
delete_pin_from_pin_group
delete_pin_groups
delete_pin_guides
delete_pipeline_net_group
delete_place_halo
delete_power_intent
delete_power_switches
delete_relative_floorplan
delete_route_blockages
delete_route_halos
delete_routes
delete_row
delete_scan_cell
delete_scan_chain
delete_scan_chain_partition
delete_sdp_rows_blockages
delete_secondary_pg_net
delete_selected_from_floorplan
delete_shields
delete_skew_groups
delete_spare_module
delete_tieoffs
delete_tracks
delete_tsv
delete_uninstantiated_modules
delete_workspace
deselectModule

deselect_obj
deselect_routes
deselect_secondary_pg_net
design_backside_bottom_routing_layer
design_backside_top_routing_layer
design_bottom_routing_layer
design_compressed_pg_db
design_cong_effort
design_dual_rail_via_pitch
design_early_clock_flow
design_express_route
design_flow_effort
design_high_freq_interposer_flow
design_ignore_followpin_vias
design_merge_trim_shapes
design_pessimistic_mode
design_power_effort
design_process_node
design_slack_weighting_method
design_tech_node
design_top_routing_layer
design_trim_grid_group
disconnect
disconnect_dangling_port
disconnect_hpin
disconnect_pin
distributed_mmmc_disable_reports_auto_redirection

E

eco_add_ports
eco_add_repeater
eco_batch_mode
eco_check_logical_equivalence
eco_clone
eco_clone_flop
eco_compare_netlist
eco_delay_calculation_effort
eco_delete_repeater
eco_design
eco_disable_change_net_name_for_flat_netlist
eco_honor_dont_touch
eco_honor_dont_use

eco_honor_fixed_status
eco_honor_fixed_wires
eco_honor_power_intent
eco_inherit_net_attribute
eco_merge_combinational_cell
eco_oa_design
eco_prefix
eco_preserve_hpin_function
eco_read_def
eco_refine_place
eco_remove_tied_inputs
eco_split_combinational_cell
eco_split_complex_flop
eco_split_flop
eco_spread_inverter
eco_swap_spare_cell
eco_update_cell
eco_update_only_layers
eco_update_timing
edit_add_polygon_point
edit_add_route_point
edit_add_trim_metal
edit_adjust_routes
edit_copy
edit_create_via
edit_cut_route
edit_duplicate_routes
edit_end_polygon_point
edit_end_route_point
edit_flow
edit_merge_routes
edit_move_routes
edit_pin
edit_resize_routes
edit_rotate
edit_split_routes
edit_split_wide_routes
edit_trim_routes
edit_update_route_layer
edit_update_route_mask
edit_update_route_net
edit_update_route_rule

edit_update_route_status
edit_update_via
edit_wire_align
edit_wire_allow_45_degree
edit_wire_arrow_incremental
edit_wire_assign_multi_pattern_color
edit_wire_auto_distribute_bus
edit_wire_auto_split_bus
edit_wire_bus_honor_start_parameters
edit_wire_bus_honor_width_setting
edit_wire_bus_total_width_horizontal
edit_wire_bus_total_width_vertical
edit_wire_change_order_at_turn
edit_wire_check_design_boundary
edit_wire_close_polygons
edit_wire_color_align_with_track
edit_wire_connect_pin
edit_wire_connect_with_specified_layer
edit_wire_create_crossover_vias
edit_wire_create_is_edit_flag
edit_wire_create_via_on_pin
edit_wire_cut_class
edit_wire_cut_wire_overlap
edit_wire_debug_file
edit_wire_delete_pin_with_wire
edit_wire_delete_tsv
edit_wire_delete_wire_via_through_layers
edit_wire_display_wire_length_with_cursor
edit_wire_draw_shield
edit_wire_draw_with_group_centerline
edit_wire_drawing_wire
edit_wire_drc_aware_cross_metal
edit_wire_drc_on
edit_wire_drc_use_non_default_spacing
edit_wire_extend_wires
edit_wire_final_check_with_verify
edit_wire_ignore_drc
edit_wire_ignore_drcedit_wire_jog_connect_layer
edit_wire_keep_floating_via
edit_wire_keep_status
edit_wire_keep_via
edit_wire_lateral_movement_range

edit_wire_layer
edit_wire_layer_horizontal
edit_wire_layer_max
edit_wire_layer_min
edit_wire_layer_vertical
edit_wire_look_down_layers
edit_wire_look_up_layers
edit_wire_max_pointer_number
edit_wire_mirror_bus_route
edit_wire_nets
edit_wire_no_merge_special_wire
edit_wire_only_show_edit_layer
edit_wire_orthogonal_connection_only
edit_wire_outer_shield_spacing
edit_wire_outer_shield_width
edit_wire_override
edit_wire_partial_overlap_threshold
edit_wire_pull_back_distance
edit_wire_reshape
edit_wire_return_obj_pointer
edit_wire_rule
edit_wire_search_route_rule_vias_only
edit_wire_shape
edit_wire_shield
edit_wire_shield_look_down_layers
edit_wire_shield_look_up_layers
edit_wire_shield_shape
edit_wire_shielding_nets
edit_wire_show_drc_info_for_edit_shape
edit_wire_sibling_look_down_layers
edit_wire_sibling_look_up_layers
edit_wire_snap
edit_wire_snap_align_to
edit_wire_snap_bus_to_pin
edit_wire_snap_end_to
edit_wire_snap_objects_to_track
edit_wire_snap_to
edit_wire_snap_to_track_honor_color
edit_wire_snap_trim_metal_to_trim_grid
edit_wire_spacing
edit_wire_spacing_horizontal
edit_wire_spacing_vertical

edit_wire_status
edit_wire_stop_at_drc
edit_wire_stretch_end
edit_wire_stretch_with_intersection
edit_wire_sub_class
edit_wire_turn_at
edit_wire_type
edit_wire_unrestricted_regular_wire_width
edit_wire_update_shield_net
edit_wire_use_fix_via
edit_wire_use_interleaving_wire_group
edit_wire_use_wire_group
edit_wire_use_wire_group_bits
edit_wire_use_wire_group_reinforcement
edit_wire_use_wire_group_reinforcement_group_via
edit_wire_use_wire_group_reinforcement_spacing
edit_wire_verbose
edit_wire_via_allow_geometry_drc
edit_wire_via_auto_snap
edit_wire_via_auto_update
edit_wire_via_cell_name
edit_wire_via_columns
edit_wire_via_create_by
edit_wire_via_cut_layer
edit_wire_via_exclude_spec
edit_wire_via_override_spec
edit_wire_via_rows
edit_wire_via_scale_height
edit_wire_via_scale_width
edit_wire_via_snap_honor_color
edit_wire_via_snap_to_intersection
edit_wire_via_type
edit_wire_width
edit_wire_width_horizontal
edit_wire_width_vertical
edit_wire_wire_override_spec
encrypt
encrypt_thermal_file
end_eco
end_flow
end_parallel_edit
end_sai

etric_capture_design_image_route_drc
eval_legacy
extract_keep_fill_wires
extract_package
extract_rc
extract_rc_assume_metal_fill
extract_rc_cap_filter_mode
extract_rc_cerebrus_license_only
extract_rc_compress_rcdb
extract_rc_coupled
extract_rc_coupling_cap_threshold
extract_rc_def_via_cap
extract_rc_effort_level
extract_rc_engine
extract_rc_extra_cmd_file
extract_rc_hard_block_obs
extract_rc_incremental
extract_rc_layer_independent
extract_rc_lef_tech_file_map
extract_rc_local_cpu
extract_rc_pvs_fill
extract_rc_qrc_cmd_file
extract_rc_qrc_cmd_type
extract_rc_qrc_output_mode
extract_rc_qrc_run_mode
extract_rc_qrc_stream_map_file
extract_rc_relative_cap_threshold
extract_rc_shrink_factor
extract_rc_signoff_stream_layer_map
extract_rc_total_cap_threshold
extract_rc_tquantus_model_file
extract_rc_tsv_subckt_file
extract_rc_turbo_reduce
extract_rc_use_qrc_oa_interface
extract_rc_use_shielding_in_detail_mode
extract_rc_via_cap

F

filter_collection
finish_floorplan
finish_floorplan_abutted_edge_as_non_row_area
finish_floorplan_active_objs

finish_floorplan_add_blockage_direction
finish_floorplan_drc_region_objs
finish_floorplan_override
fix_ac_limit_violations
fix_boundary_overlaps
fix_drc_with_patch
fix_feedthru_assign
fix_floorplan
fix_multi_drivers
fix_open_fill
fix_via
flatten_cover_cell
flatten_ilm
flatten_partitions
flexible_htrees
flip_chip_allow_layer_change
flip_chip_allow_routed_bump_edit
flip_chip_auto_pairing_file
flip_chip_bottom_layer
flip_chip_bump_use_octagon_shape
flip_chip_check_bump_access_directions
flip_chip_compaction
flip_chip_connect_power_cell_to_bump
flip_chip_constraint_file
flip_chip_drop_via_on_all_shapes
flip_chip_drop_via_on_power_mesh
flip_chip_extra_config
flip_chip_finger_direction
flip_chip_finger_max_width
flip_chip_finger_min_width
flip_chip_finger_target_mesh_layer_range
flip_chip_ignore_pad_type_check
flip_chip_lower_layer_prevent_diagonal_routing
flip_chip_lower_layer_route_width
flip_chip_multi_pad_routing_style
flip_chip_multiple_connection
flip_chip_pg_mesh_direction
flip_chip_pg_mesh_main_width
flip_chip_pg_mesh_max_width
flip_chip_pg_mesh_width
flip_chip_prevent_diagonal_wire_access_bumps
flip_chip_prevent_via_under_bump

flip_chip_prevent_via_under_bump_extension
flip_chip_route_pg_style
flip_chip_route_style
flip_chip_route_width
flip_chip_top_layer
flip_chip_via_abut_bump
flip_chip_via_to_pad_honor_min_spacing
flip_chip_wire_to_pad_honor_min_spacing
flip_or_rotate_obj
flip_or_rotate_obj
floorplan_auto_sync_master_clone
floorplan_cells_for_extra_sites
floorplan_check_types
floorplan_cut_off_place_blockage_outside_die
floorplan_cut_off_route_blockage_outside_die
floorplan_default_blockage_name_prefix
floorplan_default_power_domain_site
floorplan_enable_rectilinear_design
floorplan_extra_sites
floorplan_finfet_inst_grid
floorplan_finfet_manufacturing_grid
floorplan_finfet_placement_grid
floorplan_force_honor_snap_rule
floorplan_forced_default_tech_site
floorplan_include_io_when_init_area
floorplan_initial_all_compatible_core_site_rows
floorplan_inst_grid
floorplan_keep_rows_when_moving_power_domain
floorplan_layer_track_grid
floorplan_manufacturing_grid
floorplan_max_io_height
floorplan_minimum_sites
floorplan_move_macros_with_constraint
floorplan_move_preplaced_std_cell_only
floorplan_move_std_cell_with_constraint
floorplan_narrow_channel_threshold
floorplan_no_cut_row
floorplan_placement_grid
floorplan_power_rail_layer
floorplan_row_site_height
floorplan_row_site_width
floorplan_snap_all_corners_to_grid

floorplan_snap_block_grid
floorplan_snap_constraint_grid
floorplan_snap_core_grid
floorplan_snap_die_grid
floorplan_snap_io_grid
floorplan_snap_place_blockage_grid
floorplan_snap_place_blockage_type
floorplan_vertical_row
flow_branch
flow_caller_data
flow_db_directory
flow_error_errorinfo
flow_error_message
flow_error_write_db
flow_exclude_time_for_init_flow
flow_exit_when_done
flow_feature_values
flow_features
flow_footer_tcl
flow_header_tcl
flow_hier_path
flow_history
flow_log_directory
flow_log_prefix_generator
flow_mail_on_error
flow_mail_to
flow_metrics_file
flow_metrics_snapshot_parent_uuid
flow_metrics_snapshot_uuid
flow_overwrite_db
flow_plugin_names
flow_plugin_steps
flow_post_db_overwrite
flow_remark
flow_report_directory
flow_run_tag
flow_schedule
flow_starting_db
flow_startup_directory
flow_status_file
flow_step_canonical_current
flow_step_check_tcl

flow_step_current
flow_step_last
flow_step_last_msg
flow_step_last_status
flow_step_next
flow_summary_tcl
flow_template_feature_definition
flow_template_tools
flow_template_type
flow_template_version
flow_top
flow_user_templates
flow_verbose
flow_working_directory
flow_yamllint_exec
flowtool
flowtool_exit_timeout
flowtool_extra_arguments
flowtool_metrics_qor_excel
flowtool_metrics_qor_html
flowtool_metrics_qor_text
flowtool_metrics_qor_vivid
flowtool_predict_full_names
flowtool_summary_tcl

G

generate_esd_rlrp_report
generate_special_via_accuracy_effort
generate_special_via_add_pin_to_pin_vias
generate_special_via_align_merged_stack_via_metal
generate_special_via_allow_via_expand
generate_special_via_allow_wire_shape_change
generate_special_via_area_only
generate_special_via_check_signal_routes
generate_special_via_create_double_row_cut_via
generate_special_via_create_max_row_cut_via
generate_special_via_cut_class_preference
generate_special_via_disable_via_merge
generate_special_via_enable_check_drc
generate_special_via_extend_out_wire_end
generate_special_via_full_cut_via_only
generate_special_via_hookup_contact_max

generate_special_via_hookup_contact_pg_track
generate_special_via_hookup_fixed_grid
generate_special_via_hookup_min_distance
generate_special_via_hookup_rail_pair
generate_special_via_hookup_via_distance
generate_special_via_hookup_via_rule
generate_special_via_hookup_via_style
generate_special_via_hookup_virtual_trim_grid
generate_special_via_ignore_design_boundary
generate_special_via_ignore_drc
generate_special_via_ignore_rule_enclosure
generate_special_via_inherit_wire_status
generate_special_via_keep_existing_via
generate_special_via_keep_fixed_via
generate_special_via_mincut_preference
generate_special_via_opt_cross_via
generate_special_via_opt_via_on_routing_track
generate_special_via_parameterized_via_only
generate_special_via_partial_overlap_threshold
generate_special_via_preferred_vias_only
generate_special_via_prefix
generate_special_via_reference_boundary
generate_special_via_respect_stdcell_geometry
generate_special_via_respect_wire_ndr
generate_special_via_rule_preference
generate_special_via_set_via_expand_dir
generate_special_via_snap_via_center_to_grid
generate_special_via_split_long_via_global_grid
generate_special_via_symmetrical_via_only
generate_special_via_use_track_offset
generate_special_via_use_trim_metal_enclosure
getVersion
getVersion
get_all_layers
get_arcs
get_black_box_area
get_bond_pad
get_capacitance_unit
get_ccopt_property_common_ui_map
get_cells
get_clock_network_objects
get_clock_tree_capacitance

get_clock_tree_effective_max_capacitance
get_clock_tree_transition_time
get_clocks
get_clone_partition_orient
get_common_ui_map
get_compress_level
get_computed_shapes
get_cpf_user_attributes
get_cts_graph_traversal
get_current_constraint_mode
get_db
get_db_display_limit
get_default_switching_activity
get_designs
get_distribute_host
get_draw_view
get_dynamic_power_simulation
get_edge
get_equivalent_cells
get_feature
get_flow_config
get_generated_clocks
get_ilm_type
get_integration_route_constraint
get_io_flow_flag
get_layer_preference
get_legacy
get_lib_arcs
get_lib_cells
get_lib_clock_tree_path_delay
get_lib_pg_pins
get_lib_pins
get_libs
get_message
get_metal_fill_spacing_table
get_metric
get_metric_alias
get_metric_config
get_metric_definition
get_metric_header
get_module_view
get_nets

get_oa_default_rule_lib
get_oa_design_data
get_obj_in_area
get_partition_pin_status
get_path_groups
get_pg_fill_config
get_pg_nets
get_pg_pins
get_pg_pins_use_signal_route
get_physical_info
get_pin_constraint
get_pin_port_number
get_pins
get_ports
get_power
get_propagated_clock
get_property
get_proto_model
get_rect
get_route_attributes
get_scheduling_file
get_signoff_verify_design_config
get_skew_group_delay
get_skew_group_path
get_socv_rc_variation_factor
get_socv_reporting_nsigma_multiplier
get_time_unit
get_track_fill_config
get_transform_shapes
get_upf_port_attr
get_valid_power_switch_location
get_visible_nets
group
group_instance_suffix
group_path
gui_add_marker
gui_add_ui
gui_attach_to_cursor
gui_bind_key
gui_browse_db
gui_clear_bus_guide_colors
gui_clear_highlight

gui_clear_hinst_colors
gui_clear_scan_chains
gui_clear_spare_cells
gui_clear_trace_flightline
gui_close_cell_view
gui_create_floorplan_snapshot
gui_create_user_bundle_net
gui_ctd_close
gui_ctd_get_title
gui_ctd_get_window_id
gui_ctd_highlight
gui_ctd_open
gui_ctd_set_title
gui_ctd_write_histogram
gui_ctd_write_view
gui_ctd_zoom
gui_delete_objs
gui_delete_ui
gui_delete_user_bundle_net
gui_deselect
gui_dim_foreground
gui_find_ui
gui_fit
gui_get_box
gui_get_coord
gui_get_coords
gui_get_record_obj_by_info
gui_get_ui
gui_get_visible_bumps
gui_get_visible_net_groups
gui_group_hinst
gui_gtd_highlight_timing_report
gui_hide
gui_highlight
gui_highlight_feedthru_nets
gui_highlight_pin
gui_highlight_proto_models
gui_list_marker
gui_move_cursor
gui_open_cell_view
gui_open_gtd
gui_open_schematic

gui_pan
gui_pan_center
gui_pan_page
gui_redo
gui_redraw
gui_register_edit_callback
gui_remove_marker
gui_report_trace_flightline
gui_schematic_load_view
gui_schematic_save_view
gui_select
gui_select_highlighted
gui_set_draw_view
gui_set_power_rail_display
gui_set_power_rail_layers_nets
gui_set_select_hinst_color
gui_set_tool
gui_set_ui
gui_set_visible_bumps
gui_set_visible_net_groups
gui_set_visible_nets
gui_show
gui_show_buffer_tree
gui_show_bump_connections
gui_show_bus_guide_multi_colors
gui_show_edge_number
gui_show_floorplan_snapshot
gui_show_hinst_in_floorplan
gui_show_markers
gui_show_obj_connectivity
gui_show_partition_wire_cross_over
gui_show_scan_chains
gui_show_spare_cells
gui_show_user_bundle_nets
gui_trace_flightline
gui_undo
gui_ungroup_hinst
gui_unregister_edit_callback
gui_verbose
gui_view_box
gui_write_flightline
gui_write_picture

gui_zoom

H

help
highlight_sync_obj

I

identify_physical_power_domains
identify_proto_model
ignore_max_via_stack_rule
ilm_filter_internal_path
ilm_keep_async
ilm_keep_flatten
ilm_keep_high_fanout_ports
ilm_keep_inst_in_sdc
ilm_keep_loopback
ilm_max_num_insts
ilm_max_num_registers
ilm_slack_driven
import_module_model
init_check_netlist
init_check_output_pin_constant
init_core_rows
init_delete_floating_hnets
init_design
init_design_netlist_type
init_design_uniquify
init_eco
init_flow
init_ground_nets
init_ignore_pg_pin_polarity_check
init_keep_empty_modules
init_lef_files
init_min_dbu_per_micron
init_ml
init_mmmc_files
init_no_new_assigns
init_oa_abstract_views
init_oa_default_rule
init_oa_design_cell
init_oa_design_lib
init_oa_design_view

init_oa_layout_views
init_oa_ref_libs
init_oa_search_libs
init_oa_special_rule
init_oa_tech_lib
init_power_intent_files
init_power_nets
init_read_netlist_allow_port_mismatch
init_read_netlist_allow_verilog_error
init_read_netlist_files
init_sync_relative_path
init_timing_enabled
innovus
is_attribute
is_common_ui_mode
is_feature
is_flow

L

layer_name_no_abbreviation
legalize_floorplan
legalize_pins
list_libraries
lminus
log_file
log_verbose_style
log_verbose_type
logv_file

M

man
map_dies
merge_clock_tree_instances
merge_model_timing
merge_timing_report
metric_advanced_url_endpoint
metric_capture_depth
metric_capture_design_image
metric_capture_design_image_power_intent
metric_capture_max_drc_markers
metric_capture_min_count
metric_capture_overwrite

metric_capture_pba_tns_histogram
metric_capture_per_view
metric_capture_reg2reg_metrics
metric_capture_timing_path_groups
metric_capture_timing_paths
metric_capture_tns_histogram
metric_capture_tns_histogram_buckets
metric_capture_tns_histogram_max_slack
metric_capture_tns_histogram_paths
metric_capture_vth_per_power_domain
metric_category_default
metric_current_run_id
metric_enable
metric_summary_metrics
monitor_distributed_hosts
move_group_pins
move_macro_in_hinsts
move_obj
move_power_switches

N

nagelfar

O

oa2lef
oa_allow_analysis_only
oa_allow_bit_connection
oa_allow_tech_update
oa_bindkey_file
oa_cell_view_dir
oa_convert_diagonal_path_to
oa_cut_rows
oa_display_resource_file
oa_display_resource_file_in_library
oa_drc_fill_purpose
oa_enclose_quick_abstract_pins
oa_full_layer_list
oa_full_path
oa_inst_placed_if_none
oa_lib_create_mode
oa_lock
oa_logic_only_import

oa_merge_trim
oa_new_lib_compress_level
oa_pin_purpose
oa_push_pin_constraint
oa_quick_abstract_for_custom_cells
oa_read_system_reserved_layers
oa_silently_ignore_unsupported_vias
oa_text_purpose
oa_tie_net
oa_update_mode
oa_use_virtuoso_bindkey
oa_use_virtuoso_color
oa_view_sub_type
oa_write_cds_fixed_vias
oa_write_mask_data_locked
oa_write_net_voltage
oa_write_relative_path
opt_activity_refresh_args
opt_add_always_on_feed_through_buffers
opt_add_insts
opt_add_ports
opt_add_repeater_report_failure_reason
opt_all_end_points
opt_allow_multi_bit_on_flop_with_sdc
opt_allow_only_cell_swapping
opt_area
opt_area_recovery
opt_area_recovery_setup_target_slack
opt_clock_skew
opt_cloud_design
opt_concatenate_default_and_user_prefixes
opt_consider_routing_congestion
opt_constant_nets
opt_delete_insts
opt_design
opt_detail_drv_failure_reason
opt_detail_drv_failure_reason_max_num_nets
opt_down_size_insts
opt_drv
opt_drv_margin
opt_drv_with_miller_cap
opt_enable_data_to_data_checks

opt_enable_podv2_clock_opt_flow
opt_enable_restructure
opt_fix_fanout_load
opt_flop_pins_report
opt_flops_report
opt_hier_add_antenna_cell
opt_hier_opt_stage
opt_high_effort_cells
opt_hold_allow_overlap
opt_hold_allow_resize
opt_hold_allow_setup_tns_degradation
opt_hold_cells
opt_hold_ignore_path_groups
opt_hold_on_excluded_clock_nets
opt_hold_slack_threshold
opt_hold_target_slack
opt_honor_density_screen
opt_honor_fences
opt_leakage_to_dynamic_ratio
opt_max_density
opt_max_length
opt_move_insts
opt_multi_bit_combinational_merge_timing_effort
opt_multi_bit_combinational_mode
opt_multi_bit_combinational_opt
opt_multi_bit_combinational_split_timing_effort
opt_multi_bit_flop_merge_bank_label_inference
opt_multi_bit_flop_merge_timing_effort
opt_multi_bit_flop_name_prefix
opt_multi_bit_flop_name_separator
opt_multi_bit_flop_name_suffix
opt_multi_bit_flop_opt
opt_multi_bit_flop_reorder_bits
opt_multi_bit_flop_split_report_failure_reason
opt_multi_bit_flop_split_timing_effort
opt_multi_bit_unused_bit_count
opt_multi_bit_unused_bits
opt_new_inst_prefix
opt_new_net_prefix
opt_one_pass_lec
opt_pattern
opt_pin_swapping

opt_podv2_flow_effort
opt_post_route_allow_overlap
opt_post_route_area_reclaim
opt_post_route_art_flow
opt_post_route_check_antenna_rules
opt_post_route_drv_recovery
opt_post_route_fix_clock_drv
opt_post_route_fix_glitch
opt_post_route_fix_si_transitions
opt_post_route_hold_recovery
opt_post_route_report_si_transitions
opt_post_route_setup_recovery
opt_power
opt_power_effort
opt_pre_route_ndr_aware
opt_preserve_all_sequential
opt_preserve_hpin_function
opt_remove_redundant_insts
opt_resize_flip_flops
opt_resize_level_shifter_and_iso_insts
opt_resize_power_switch_insts
opt_sequential_genus_restructure_report_failure_reason
opt_setup_target_slack
opt_signoff
opt_signoff_add_inst
opt_signoff_add_load
opt_signoff_allow_skewing
opt_signoff_along_route_buffering
opt_signoff_buffer_cell_list
opt_signoff_check_drv_from_hold_views
opt_signoff_check_type
opt_signoff_clock_cell_list
opt_signoff_clock_max_level
opt_signoff_delete_inst
opt_signoff_disable_geometry_checks
opt_signoff_drv_margin
opt_signoff_eco_file_prefix
opt_signoff_em_type
opt_signoff_fix_cell_em
opt_signoff_fix_clock_drv
opt_signoff_fix_data_drv
opt_signoff_fix_glitch

opt_signoff_fix_hold_allow_setup_optimization
opt_signoff_fix_hold_allow_setup_tns_degrade
opt_signoff_fix_hold_with_margin
opt_signoff_fix_ir_drop
opt_signoff_fix_max_cap
opt_signoff_fix_max_transition
opt_signoff_fix_osus_glitch
opt_signoff_fix_si_slew
opt_signoff_fix_slack_in_range_lower_limit
opt_signoff_fix_slack_in_range_upper_limit
opt_signoff_fix_xtalk
opt_signoff_hold_target_slack
opt_signoff_hold_xtalk_delta_threshold
opt_signoff_hold_xtalk_slack_threshold
opt_signoff_ignore_drv_checks
opt_signoff_keep_tmp_files
opt_signoff_legal_only
opt_signoff_load_cell_list
opt_signoff_max_cap_margin
opt_signoff_max_opt_runtime
opt_signoff_max_paths
opt_signoff_max_runtime
opt_signoff_max_slack
opt_signoff_max_transition_margin
opt_signoff_num_report_paths
opt_signoff_nworst
opt_signoff_optimize_core_only
opt_signoff_optimize_replicated_modules
opt_signoff_optimize_sequential_cells
opt_signoff_partition_list_file
opt_signoff_pba_eco_route
opt_signoff_pba_effort
opt_signoff_post_mask
opt_signoff_post_mask_decap_cells
opt_signoff_post_mask_filler_cells
opt_signoff_post_mask_gate_array_filler_cells
opt_signoff_post_mask_logic_cells
opt_signoff_post_sta_tcl
opt_signoff_power_aware
opt_signoff_power_opt_focus
opt_signoff_pre_sta_tcl
opt_signoff_prefix

opt_signoff_preserve_filler
opt_signoff_read_eco_opt_db
opt_signoff_read_irdrop_db
opt_signoff_resize_inst
opt_signoff_retime
opt_signoff_retime_depth
opt_signoff_retime_depth_early
opt_signoff_retime_mode
opt_signoff_routing_congestion_aware
opt_signoff_select_drv_net_file
opt_signoff_select_drv_pin_file
opt_signoff_select_hold_endpoints
opt_signoff_select_setup_endpoints
opt_signoff_set_hold_endpoints_margin
opt_signoff_set_setup_endpoints_margin
opt_signoff_setup_recovery
opt_signoff_setup_target_slack
opt_signoff_setup_xtalk_delta_threshold
opt_signoff_setup_xtalk_slack_threshold
opt_signoff_skip_drv_net_file
opt_signoff_swap_inst
opt_signoff_verbose
opt_signoff_write_eco_opt_db
opt_skew
opt_skew_ccopt
opt_skew_delay_pre_cts
opt_skew_macro_only
opt_skew_max_allowed_delay
opt_skew_min_allowed_delay
opt_skew_no_boundary
opt_skew_post_route
opt_skew_pre_cts
opt_target_based_opt_file
opt_target_based_opt_file_only
opt_target_based_opt_hold_file
opt_tied_inputs
opt_time_design_compress_reports
opt_time_design_expanded_view
opt_time_design_num_paths
opt_time_design_report_net
opt_time_design_vertical_timing_summary
opt_unfix_clock_insts

opt_verbose
opt_virtual
ower_intent_share_well_always_on_buffering_support

P

pack_align_macros
pack_module_model_libs
parse_proc_arguments
partition_floorplan_export
partition_floorplan_import
partition_push_network
partition_push_special_wires
paste_obj
place_area_ios
place_blocks_refine
place_bond_pad
place_bond_pad_by_stagger
place_cell_edge_spacing
place_cells_at_center_for_feedthrus
place_clones
place_cloud_design
place_connected
place_design
place_design_floorplan_mode
place_design_refine_macro
place_design_refine_place
place_detail
place_detail_activity_power_driven
place_detail_allow_border_pin_abut
place_detail_allow_single_height_row_symmetry_x
place_detail_check_cut_spacing
place_detail_check_inst_space_group
place_detail_check_route
place_detail_clock_tree
place_detail_color_aware_legal
place_detail_context_aware_legal
place_detail_eco_max_distance
place_detail_eco_priority_insts
place_detail_fixed_shifter
place_detail_honor_inst_pad
place_detail_io_pin_blockage
place_detail_ir_aware_max_drive_strength

place_detail_irdrop_aware_effort
place_detail_irdrop_aware_timing_effort
place_detail_irdrop_region_number
place_detail_legalization_inst_gap
place_detail_m3_stripe_shrink
place_detail_max_shifter_column_depth
place_detail_max_shifter_depth
place_detail_max_shifter_row_depth
place_detail_no_filler_without_implant
place_detail_pad_fixed_insts
place_detail_pad_physical_cells
place_detail_preroute_as_obs
place_detail_preserve_routing
place_detail_remove_affected_routing
place_detail_sdp_alignment_in_refine
place_detail_swap_eeq_cells
place_detail_use_check_drc
place_detail_use_diffusion_transition_fill
place_detail_use_gate_array_filler_groups
place_detail_use_no_diffusion_one_site_filler
place_detail_wire_length_opt_effort
place_eco
place_fix_congestion
place_global_activity_power_driven
place_global_activity_power_driven_effort
place_global_align_macro
place_global_allow_3d_stack
place_global_auto_blockage_in_channel
place_global_clock_gate_aware
place_global_clock_power_driven
place_global_clock_power_driven_effort
place_global_cong_effort
place_global_cpg_effort
place_global_cpg_file
place_global_enable_advanced_pipeline
place_global_enable_distributed_place
place_global_ignore_scan
place_global_ignore_spare
place_global_max_density
place_global_module_aware_spare
place_global_module_padding
place_global_place_io_pins

place_global_reorder_scan
place_global_sdp_alignment
place_global_sdp_place
place_global_soft_guide_strength
place_global_timing_effort
place_global_uniform_density
place_hard_fence
place_inst
place_jtag
place_macro_detail
place_opt_design
place_opt_post_place_tcl
place_opt_run_global_place
place_pad
place_pad_ios
place_partition_clone
place_peripheral_ios
place_pipelines
place_sdp_clock_location
place_sdp_disable_extend_core
place_sdp_honor_orient
place_sdp_legalization
place_sdp_max_move_distance
place_sdp_num_column
place_sdp_place_report
place_sdp_pre_fixed_cells_blockage_direction
place_sdps
place_spare_modules
place_spare_update_timing_graph
plan_design_domain
plan_design_rail_model
plan_design_total_power
pop_snapshot_stack
power-grid library commands
power_add_simulation
power_adjust_input_activity_in_iterations
power_adjust_macro_activity_in_iterations
power_analysis_temperature
power_annotation_detail_report
power_auto_twf_delay_annotation
power_average_rise_fall_cap
power_block_independent_peakpower

power_bulk_pins
power_capacity
power_case_insensitive_mapping
power_clock_source_as_clock
power_comprehensive_automapping
power_constant_override
power_corner
power_create_driver_db
power_current_generation_method
power_db_name
power_decap_cell_list
power_default_frequency
power_default_slew
power_default_supply_voltage
power_disable_clock_gate_clipping
power_disable_leakage_scaling
power_disable_static
power_distribute_switching_power
power_distributed_combine_report_format
power_distributed_setup
power_domain_based_clipping
power_dynamic_glitch_filter
power_dynamic_scale_clock_by_frequency
power_dynamic_scale_clock_by_name
power_dynamic_vectorless_ranking_methods
power_enable_auto_mapping
power_enable_auto_queue
power_enable_disk_mapping
power_enable_duty_propagation_with_global_activity
power_enable_dynamic_current_slew_load_interpolation
power_enable_dynamic_scaling
power_enable_flop_state_propagation
power_enable_generated_clock
power_enable_input_net_power
power_enable_interactive_reports
power_enable_mt_state_propagation
power_enable_multithread_reports
power_enable_pba_for_tempus_pi
power_enable_power_target_flow
power_enable_rtl_dynamic_vector_based
power_enable_scan_report
power_enable_slew_based_ccs_pin_cap

power_enable_state_propagation
power_enable_tempus_pi
power_enable_xp
power_enhanced_blackbox_avg
power_enhanced_blackbox_max
power_equivalent_annotation
power_event_based_leakage_power
power_external_load_config_file
power_extraction_tech_file
power_extractor_include
power_fanout_limit
power_force_library_merging
power_from_x_transition_factor
power_from_z_transition_factor
power_generate_activity_mapping_report
power_generate_current_for_rail
power_generate_flop_ranking_data
power_generate_leakage_power_map_based_on_calculated_leakage
power_generate_static_report_from_state_propagation
power_grid_libraries
power_handle_glitch
power_handle_tristate
power_hier_delimiter
power_honor_combinational_logic_on_clock_net
power_honor_negative_energy
power_honor_net_activity
power_honor_non_mission_leakage
power_hybrid_analysis
power_ignore_control_signals
power_ignore_data_phase_for_clock
power_ignore_end_toggles_in_profile
power_ignore_glitches_at_same_time_stamp
power_ignore_inout_pin_cap
power_include_file
power_include_initial_x_transitions
power_include_sequential_clock_pin_power
power_include_timing_in_current_file
power_intent_allow_back_to_back_isolation
power_intent_allow_nested_default_domain
power_intent_allow_power_domain_min_gap_zero
power_intent_allow_shifter_voltage_mismatch
power_intent_assume_iso_enable_pin_is_always_on

power_intent_check_all_nets_for_domain_crossing
power_intent_do_not_use_top_domain_for_port_voltage
power_intent_honor_power_domain_for_domain_crossing_route
power_intent_honor_power_domain_for_intra_domain_route
power_intent_include_dot_lib_related_pg_pin
power_intent_use_cpf_global_connect_for_always_on_buffer
power_intent_use_cpf_global_connect_for_shifter
power_intent_use_effective_domain_for_iso_shifter_insertion
power_ir_derated_timing_view
power_keep_clock_gate_ratio_in_iterations
power_leakage_scale_factor_for_temperature
power_lib_files
power_library_preference
power_match_state_for_logic_x
power_merge_switched_net_currents
power_method
power_min_leaf_count
power_multi_scenario_simulation
power_multibit_flop_toggle_behavior
power_off_pg_nets
power_output_current_data_prefix
power_output_dir
power_partition_count
power_partition_twf
power_pin_based_twf
power_pre_simulation_empty_period
power_pre_simulation_period
power_pre_simulation_power_exclude_period
power_precision
power_quit_on_activity_coverage_threshold
power_read_rcdb
power_relax_arc_match
power_report_black_boxes
power_report_idle_instances
power_report_instance_switching_info
power_report_instance_switching_list
power_report_library_usage
power_report_missing_bulk_connectivity
power_report_missing_input
power_report_missing_nets
power_report_scan_chain_statistics
power_report_statistics

power_report_time_display_fraction_digits
power_report_twf_attributes
power_reuse_flop_ranking_data
power_reuse_flop_ranking_data_hier
power_scale_to_sdc_clock_frequency
power_scan_chain_activity
power_scan_chain_name_pattern
power_scan_control_file
power_scan_multi_bit_flop_chain_type
power_settling_buffer
power_smart_window
power_split_bus_power
power_start_time_alignment
power_state_dependent_leakage
power_stateprop_ignore_unannot_pins
power_static_multi_mode_scenario_file
power_static_netlist
power_thermal_input_file
power_thermal_leakage_temperature_scale_table_file
power_to_x_transition_factor
power_to_z_transition_factor
power_transition_factor_based_duty
power_transition_time_method
power_twf_delay_annotation
power_twf_load_cap
power_use_cell_leakage_power_density
power_use_fastest_clock_for_dynamic_scheduling
power_use_lef_for_missing_cells
power_use_physical_partition
power_use_zero_delay_vector_file
power_vector_based_multithread
power_vector_profile_mode
power_worst_case_vector_activity
power_worst_step_size
power_worst_window_count
power_worst_window_coverage
power_worst_window_reports
power_worst_window_size
power_worst_window_type
power_write_bbox
power_write_db
power_write_default_pti_files

power_write_dynamic_currents
power_write_gui_db
power_write_profiling_db
power_write_simulation_db
power_write_static_currents
power_x_count_transition_using_3_states
power_x_transition_factor
power_z_transition_factor
power_zero_delay_vector_toggle_shift
print_full_message_summary
print_message
program_major_version
program_name
program_short_name
program_version
proto_allow_model_with_io
proto_allow_power_domain_in_flexmodel
proto_create_dir
proto_create_high_fanout_ps_per_micron
proto_create_lib
proto_create_metal_fill_ndr
proto_create_metal_fill_nominal
proto_create_multi_corner_ps_per_micron
proto_create_ndr_ps_per_micron
proto_create_no_flex_filler
proto_create_partition_as_flexmodel
proto_create_pipeline_flop
proto_create_power_domain_ps_per_micron
proto_create_timing_budget
proto_design
proto_design_cover_fixed_macros
proto_design_effort
proto_design_flexmodel_constraint_type
proto_design_keep_guide
proto_design_place_macro
proto_design_remove_overlap
proto_identify_engine
proto_identify_estimated_flexmodel_number
proto_identify_exclude_module
proto_identify_exclude_module_and_parent
proto_identify_exclude_module_tree
proto_identify_honor_objects_hierarchy

proto_identify_max_inst
proto_identify_min_inst
proto_keep_inst_file_only
proto_keep_instance_defined_in_sdc
proto_keep_slack_improve_ndr
proto_max_report_ndr_net
proto_route_net_ndr
proto_timing_net_delay
proto_use_timing_net_delay_model
proto_verbose
pull_block_constraints
push_snapshot_stack
puts

Q

query_objects
query_power_data

R

Related Documents
range_collection
readTransitionFile
read_activity_map_file
read_black_box_netlist_file
read_bump_color_map_file
read_bump_locations
read_cts_config
read_db
read_db_directory
read_db_file_check
read_db_stop_at_design_in_memory
read_db_tool_name
read_db_version
read_def
read_def_check_mask_shifts
read_design_stack_config
read_die_abstract
read_drc_markers
read_eco
read_flip_chip_property
read_floorplan
read_flow

read_footprint
read_hierarchical_def
read_ilm
read_ilm_eco_db
read_io_file
read_io_update
read_markers
read_metric
read_metric_config
read_metric_html_package
read_mmmc
read_module_model
read_name_mapping
read_netlist
read_oa
read_package
read_parallel_edit_files
read_partition_pins
read_path_categories
read_path_descriptions
read_physical
read_physical_check_antenna
read_physical_check_mask_shifts
read_physical_extend_cell_obs_shapes_under_trim
read_physical_extend_polygon_cell_shapes
read_physical_merge_multi_port_to_single_port
read_physical_must_join_all_ports
read_power_intent
read_power_rail_results
read_power_switches
read_power_switches_for_floorplan
read_rc
read_safety_tmr
read_sai
read_sdf
read_sdp_file
read_spef
read_stream
read_taf
read_timing_debug_report
read_tsv_config
read_twf

read_usf
read_workspace
read_write_lef_check_uncolored_shapes
redirect
register_trigger
relink_db_files
remove_from_collection
rename_obj
rename_pg_library
rename_snapshot
reorder_power_switch_chains
reorder_scan
reorder_scan_add_scan_port_prefix
reorder_scan_clock_aware
reorder_scan_comp_logic
reorder_scan_effort
reorder_scan_enable_for_partition
reorder_scan_keep_hinst_port_name
reorder_scan_keep_hport
reorder_scan_prefer_horizontal
reorder_scan_prefer_vertical
reorder_scan_skip_mode
reorder_scan_swap_effort
report_always_on_buffer
report_analysis_coverage
report_analysis_views
report_annotated_assertions
report_annotated_check
report_annotated_delay
report_annotated_parasitics
report_aocv_derate
report_area
report_bundled_bus
report_case_analysis
report_ccopt_skew_groups
report_cell_edge_spacing
report_cell_edge_type
report_cell_instance_timing
report_cell_padding
report_cell_stack_area
report_cell_stack_group
report_cell_virtual_align

report_clock_gating_check
report_clock_timing
report_clock_tree_convergence
report_clock_tree_halo_violations
report_clock_tree_structure
report_clock_trees
report_clocks
report_congested_area
report_congestion
report_constraint
report_cppr
report_critical_insts
report_critical_nets
report_critical_pins
report_cts_cell_filtering_reasons
report_cts_cell_name_info
report_dangling_ports
report_decap_cell_candidates
report_decaps
report_delay_calculation
report_density_in_area
report_density_map
report_design
report_differential_voltage
report_disk
report_esd_network
report_esd_voltage
report_fanin
report_fanout
report_floating_nets
report_floorplan_obj
report_flow
report_gate_count
report_gates
report_gui_edit_callback
report_hidden_usage
report_ignored_nets
report_ilm_status
report_inactive_arcs
report_inst_padding
report_inst_power
report_inst_space_group

report_instance_cdb
report_instance_library
report_irdrop_regions
report_isolation
report_jtag_inst
report_lef_equivalent_cells
report_legal_wire_width_for_bump
report_length_violations
report_lib_arcs
report_markers
report_messages
report_metal_fill
report_metal_stack
report_metric
report_min_pulse_width
report_mode
report_module_model
report_multi_bit_ffs
report_multibit
report_narrow_channel
report_net
report_net_groups
report_net_parasitics
report_net_wires
report_netlist_statistics
report_noise
report_noise_lib_pin
report_noise_margin
report_noise_propagation
report_oa_lib
report_obj
report_obj_connectivity
report_obj_display_limit
report_object_property_names
report_off_grid_blocks
report_package
report_partition_boundary_ports
report_partitions
report_path_group_options
report_path_groups
report_pg_keep_out
report_pin_assignment_statistics

report_pin_density
report_pin_density_map
report_pin_density_map_display_step
report_pin_density_map_grid_size
report_pin_density_map_grid_unit
report_pin_density_map_threshold
report_pin_insertion_delays
report_pipelines
report_place_density_map_grid_size
report_place_density_map_grid_unit
report_place_density_map_ignore_block
report_place_density_map_ignore_filler
report_place_density_map_threshold
report_ports
report_power_domains
report_power_rail_results
report_power_route
report_power_switches
report_preserves
report_probe_pins
report_property
report_proto_models
report_qor
report_rail
report_rc
report_rc_factors
report_rcdb
report_resource
report_route
report_route_type_constraints
report_sai_constraint
report_scan_cells
report_scan_chain
report_scan_chain_partitions
report_seed_connections
report_sequential
report_shields
report_shifters
report_signal_resistance
report_skew_groups
report_socv_interconnect_variation
report_socv_library

report_special_routes
report_statistical_timing_derate_factors
report_summary
report_threshold_instance_count
report_timing
report_timing_budget
report_timing_budget_exception
report_timing_derate
report_timing_model
report_timing_summary
report_tracks
report_two_pin_chain
report_unaligned_nets
report_unannotated_nets
report_unit_parasitics
report_via_def
report_voltage
report_voltage_scaling
report_voltage_swing
report_wire_load
report_wire_path
report_wires
reset_annotated_check
reset_annotated_delay
reset_annotated_transistion
reset_annotated_transition
reset_aocv_stage_weight
reset_bumps_fixed_assign
reset_case_analysis
reset_clock
reset_clock_gating_check
reset_clock_groups
reset_clock_latency
reset_clock_sense
reset_clock_transition
reset_clock_tree_latency
reset_clock_tree_routing_state
reset_clock_uncertainty
reset_cts_config
reset_data_check
reset_db
reset_decap_cell_candidates

reset_design
reset_disable_clock_gating_check
reset_disable_timing
reset_driving_cell
reset_fixed_bond_pad
reset_fixed_module_size
reset_flow
reset_generated_clock
reset_glitch_derate
reset_ideal_latency
reset_ideal_network
reset_ideal_network
reset_ilm
reset_input_delay
reset_instance_library
reset_jtag
reset_load
reset_max_capacitance
reset_max_fanout
reset_max_time_borrow
reset_max_transition
reset_metric_config
reset_metric_run_id
reset_min_capacitance
reset_min_fanout
reset_min_pulse_width
reset_min_transition
reset_mode
reset_modified_budget
reset_noise_lib_pin
reset_output_delay
reset_parasitics
reset_partition_pins
reset_path_adjust
reset_path_adjust_group
reset_path_exception
reset_path_group
reset_path_group_options
reset_pin_constraintsUsage
reset_power_activity
reset_probe_pin
reset_propagated_clock

reset_property
reset_pulse_clock_max_width
reset_pulse_clock_min_transition
reset_resistance
reset_sdf_assertions
reset_selective_blockage_gate
reset_sense
reset_spare_insts
reset_timing_derate
reset_usf
reset_wire_load_mode
reset_wire_load_model
resize_floorplan
resize_floorplan_channel
resize_floorplan_cong_aware
resize_floorplan_honor_halo
resize_floorplan_io_fix
resize_floorplan_io_move_with_edge
resize_floorplan_io_relative
resize_floorplan_maintain_resource_ratio
resize_floorplan_relative
resize_floorplan_shift_based
resize_floorplan_shrink_fence
resize_floorplan_snap_to_track
resize_partition_to_abut
resume
route_add_antenna_inst_prefix
route_add_shields
route_adjust_auto_via_weight
route_allow_inst_overlaps
route_allow_pin_as_feedthru
route_antenna_cell_name
route_antenna_diode_insertion
route_clock_tree_nets
route_concurrent_minimize_via_count_effort
route_connect_to_bumps
route_design
route_detail
route_detail_add_passive_fill_only_on_layers
route_detail_allow_passive_fill_only_in_layers
route_detail_antenna_eco_list_file
route_detail_auto_stop

route_detail_check_mar_on_cell_pin
route_detail_end_iteration
route_detail_fix_antenna
route_detail_fix_antenna_on_secondary_pg_nets
route_detail_fix_antenna_with_gate_array_filler_mode
route_detail_merge_abutting_cut
route_detail_min_length_for_spread_wire
route_detail_min_length_for_widen_wire
route_detail_min_slack_for_opt_wire
route_detail_no_taper_in_layers
route_detail_no_taper_on_output_pin
route_detail_on_grid_only
route_detail_post_route_litho_repair
route_detail_post_route_spread_wire
route_detail_post_route_swap_via
route_detail_post_route_via_pillar_effort
route_detail_post_route_wire_widen
route_detail_post_route_wire_widen_rule
route_detail_postroute_via_priority
route_detail_search_and_repair
route_detail_signoff_effort
route_detail_stub_routing_in_first_layer
route_detail_use_multi_cut_via_effort
route_diode_insertion_for_clock_nets
route_early_global
route_early_global_effort_level
route_early_global_honor_partition_allow_feedthru
route_early_global_honor_partition_fence
route_early_global_honor_partition_pin
route_early_global_honor_partition_pin_guide
route_early_global_honor_power_domain
route_early_global_horizontal_supply_scale_factor
route_early_global_num_tracks_per_clock_wire
route_early_global_reverse_direction_regions
route_early_global_route_selected_net_only
route_early_global_secondary_pg
route_early_global_secondary_pg_max_fanout
route_early_global_stripe_layer_range
route_early_global_vertical_supply_scale_factor
route_eco
route_eco_ignore_existing_route
route_enable_route_rule_si_limit_length

route_enforce_route_rule_on_special_net_wire
route_extra_via_enclosure
route_fix_clock_nets
route_fix_signoff_drc
route_flexible_htrees
route_flip_chip
route_global_detail
route_hierarchical_clock
route_high_freq_constraint_groups
route_high_freq_match_report_file
route_high_freq_num_reserved_layers
route_high_freq_remove_floating_shield
route_high_freq_search_repair
route_high_freq_shield_trim_length
route_honor_exclusive_region
route_honor_power_domain
route_ignore_antenna_top_cell_pin
route_ignore_follow_pin_shapes
route_interposer_allow_diagonal_trunk
route_interposer_interlayer_shielding_layers
route_interposer_interlayer_shielding_nets
route_interposer_interlayer_shielding_offsets
route_interposer_interlayer_shielding_widths
route_interposer_same_layer_shielding_net
route_interposer_same_layer_shielding_width_spacing
route_interposer_trunk_routing_layers
route_interposer_trunk_routing_width_spacing
route_number_fail_limit
route_number_thread
route_number_warning_limit
route_opt_design
route_pg
route_point_to_point
route_process_node
route_rc_extraction_corner
route_relaxed_route_rule_spacing_to_power_ground_nets
route_reserve_space_for_multi_cut
route_reverse_direction
route_route_clock_nets_first
route_secondary_pg_pins
route_selected_net_only
route_shield_crosstie_offset

route_shield_report_skip_status
route_shield_tap_cell_insertion
route_shield_tap_cell_name
route_skip_analog
route_special
route_special_allow_non_preferred_direction_route
route_special_avoid_over_core_row_layer
route_special_block_pin_connect_ring_pin_corners
route_special_block_pin_route_with_pin_width
route_special_connect_broken_core_pin
route_special_core_pin_ignore_obs
route_special_core_pin_length
route_special_core_pin_length_as_inst
route_special_core_pin_max_via_scale
route_special_core_pin_merge_limit
route_special_core_pin_refer_to_follow_pin
route_special_core_pin_reference_macro
route_special_core_pin_site_rail_width
route_special_core_pin_snap_to
route_special_core_pin_stop_route
route_special_endcap_as_core
route_special_extend_nearest_target
route_special_jog_threshold_ratio
route_special_layer_non_preferred_direction_cost
route_special_layer_preferred_direction_cost
route_special_pad_pin_min_via_size
route_special_pad_pin_split
route_special_pad_ring_use_lef
route_special_pg_pin_as_signal
route_special_secondary_pin_max_gap
route_special_secondary_pin_rail_width
route_special_signal_pin_as_pg
route_special_split_long_via
route_special_target_number
route_special_target_search_distance
route_special_time_limit
route_special_via_connect_to_shape
route_special_via_through_to_closest_ring
route_special_welltap_as_endcap
route_strict_honor_route_rule
route_strictly_honor_1d_routing
route_stripe_layer_range

route_third_party_data
route_tieoff_to_shapes
route_trial_max_print_ignored_pad_nets
route_trim_pull_back_distance_from_boundary
route_trunk_with_cluster_target_size
route_use_auto_via
route_via_weight
route_with_eco
route_with_litho_driven
route_with_si_driven
route_with_timing_driven
route_with_trim_metal
route_with_via_in_pin
route_with_via_only_for_block_cell_pin
route_with_via_only_for_stdcell_pin
run_abstract
run_abstract_abstract_blockage_cut_around_pin
run_abstract_antenna_connectivity
run_abstract_antenna_diffusion_geometry
run_abstract_antenna_gate_geometry
run_abstract_blockage_detailed_layers
run_abstract_boundary_layers
run_abstract_cell_symmetry
run_abstract_export_lef_version
run_abstract_extract_layers_power
run_abstract_extract_layers_signal
run_abstract_extract_pin_layers_power
run_abstract_extract_pin_layers_signal
run_abstract_input_cell_type
run_abstract_input_lef_tech_file
run_abstract_input_stream_layer_map_file
run_abstract_keep_output_files
run_abstract_pins_analog_names
run_abstract_pins_clock_names
run_abstract_pins_ground_names
run_abstract_pins_output_names
run_abstract_pins_power_names
run_abstract_pins_text_pin_map
run_abstract_pre_abstract_script
run_abstract_selected_cells
run_abstract_site_name
run_abstract_verbose

run_clp
run_flow
run_metric_category
run_vsr

S

safety_flow_enable
safety_midas_enable
safety_ser_cells
safety_tmr_isolate_clock
safety_tmr_spacing
safety_tmr_spacing_x
safety_tmr_spacing_y
scale_what_if_capacitance
scale_what_if_current
scale_what_if_resistance
schedule_flow
script_search_path
select_obj
select_routes
select_secondary_pg_net
selected
setObjFPlanBox
setObjFPlanBox
setObjFPlanBox
setObjFPlanBox
setObjFPlanBox
setObjFPlanBox
setViaGenMode
setViaGenMode
setViaGenMode
set_advanced_package_options
set_advanced_pg_library_mode
set_advanced_rail_options
set_all_ios_to_fixed
set_analysis_view
set_annotated_check
set_annotated_delay
set_annotated_glitch
set_annotated_transition
set_aocv_interface_path_offset
set_aocv_stage_weight

set_aocv_thresholds
set_blackbox_master_orient
set_blackbox_master_to_r0
set_bond_pads_to_fixed
set_bottom_io_pad_orient
set_buffer_footprint
set_bump_master
set_bumps_fixed_assign
set_bumps_place_status
set_case_analysis
set_cdb_binding
set_cell_edge_spacing
set_cell_edge_type
set_cell_padding
set_cell_power_domain
set_cell_stack_area
set_cell_stack_group
set_clock_exclusivity
set_clock_gating_check
set_clock_groups
set_clock_sense
set_clock_transition
set_clock_uncertainty
set_clock_value
set_clone_partition_orient
set_compress_level
set_compress_level
set_current_design
set_cycle_budget_ratio
set_data_check
set_db
set_db_verbose
set_default_view
set_default_workspace
set_delay_footprint
set_die_model
set_disable_clock_gating_check
set_disable_timing
set_distributed_hosts
set_dont_touch
set_dont_touch_network
set_dont_use

set_drive
set_driving_cell
set_dynamic_power_simulation
set_dynamic_rail_simulation
set_extract_shrink_factor
set_false_path
set_feature
set_fixed_budget
set_flipping
set_floorplan_row_spacing_and_type
set_flow_config
set_flowkit_read_db_args
set_flowkit_write_db_args
set_glitch_derate
set_ideal_latency
set_ideal_network
set_ideal_transition
set_ilm_type
set_input_delay
set_input_transition
set_inst_group_physical_hier
set_inst_padding
set_instance_library
set_instance_placement_status
set_instance_temperature_file
set_integration_route_constraint
set_inverter_footprint
set_io_constraints
set_io_flow_flag
set_io_row_margin
set_io_thresholds
set_isolated_cut_rule
set_jtag
set_jtag_to_unplaced
set_layer_preference
set_library_unit
set_license_check
set_limited_access_feature
set_load
set_lockup_element
set_logic_one
set_logic_zero

set_max_cap_per_freq
set_max_cap_per_freq_transition
set_max_capacitance
set_max_delay
set_max_fanout
set_max_route_layer
set_max_time_borrow
set_max_transition
set_max_transition_per_freq
set_message
set_metal_fill
set_metal_fill_spacing_table
set_metric
set_metric_alias
set_metric_header
set_min_capacitance
set_min_delay
set_min_fanout
set_min_pulse_width
set_min_transition
set_mode
set_model_priority
set_module_model
set_module_size_fixed
set_module_view
set_multi_cpu_usage
set_multi_die_analysis_mode
set_multicycle_path
set_mutex_attacker
set_net_group
set_noise_lib_pin
set_normalized_driver_waveform_lib
set_oa_cell_binding
set_object_color
set_off_chip_package_trace
set_outbound_report
set_output_delay
set_package
set_partition_area_io
set_partition_pin_status
set_partition_pin_use
set_partition_user_constraints_file

set_path_adjust
set_path_adjust_group
set_path_group_options
set_pg_fill_config
set_pg_library_mode
set_pg_nets
set_pg_pins_use_signal_route
set_pin_constraint
set_power_calculation_temperature
set_power_data
set_power_include_file
set_power_lib_files
set_power_output_dir
set_power_pads
set_preference
set_probe_pin
set_proc_verbose
set_promoted_macro_pin
set_propagated_clock
set_property
set_proto_model
set_proto_model_physical_constraint
set_proto_timing_settings
set_pulse_clock_max_transition
set_pulse_clock_max_width
set_pulse_clock_min_transition
set_pulse_clock_min_width
set_quiet_aggressor
set_rail_analysis_config
set_resistance
set_resize_line
set_route_attributes
set_route_blockage_default_layer
set_safety_mechanism_rules_dcls
set_safety_mechanism_rules_tmr
set_scan_cell
set_scheduling_file
set_sdp_origin
set_selective_blockage_gate
set_sense
set_snap_grid
set_socv_constraint_config

set_socv_rc_variation_factor
set_socv_reporting_nsigma_multiplier
set_spare_insts
set_special_route_type
set_table_style
set_timing_derate
set_track_fill_config
set_twf_attribute
set_value
set_via_fill
set_virtual_clock_network_parameters
set_voltage_regulator_module
set_window_preference
set_wire_load_mode
set_wire_load_model
set_wire_load_selection_group
show_ccopt_cell_name_info
skew_groups
snap_floorplan
snap_floorplan_io
soft_stack_size_limit
sort_collection
sort_db
source
source_continue_on_error
source_echo_filename
source_verbose
source_verbose_line_length_limit
space_bond_pads
space_selected
specify_cell_virtual_align
split_rectilinear_inst
split_row
start_parallel_edit
stretch_rows
summaryReport
suspend
swapRegularCellWithAlwaysOnCell
swap_bump_assignment
swap_pins
swap_well_taps
switching_power_on_rise_only

synthesize_flexible_htrees

T

tcl_error_info
tcl_partial_cmd_argument_matching
tcl_return_display_length_limit
tech_db_units
tech_finfet_grid_direction
tech_finfet_grid_offset
tech_finfet_grid_pitch
tech_inst_mask_shift_layers
tech_mfg_grid
time_design
time_design_signoff
time_info
timing_apply_setup_hold_exceptions_to_data_check_related_pin
timing_check_timing_signal_level_high_to_low_threshold
timing_check_timing_signal_level_low_to_high_threshold
timing_constraint_enable_separate_multicycle_data_checks
timing_constraint_path_delay_exclude_io_delay_from_ignore_clock_latency
timing_enable_latch_thru_mode
timing_library_infer_socv_from_aocv
timing_report_disable_max_paths_per_group
timing_report_drv_enable_clock_source_as_clock
timing_report_socv_summary_mean_sigma
timing_socv_preserve_variation_with_annotations
timing_use_verilog_for_model_netlist
timing_waveform_aware_pulse_width_checks_low_voltage_level
trace_jtag
trace_obj_connectivity
trace_obj_macro_pins
trace_obj_max_fanin_fanout
trace_obj_register_inputs
trace_obj_register_outputs
trim_design
trim_metal_fill
trim_pg
trim_pg_library

U

ui_precision
ui_precision_capacitance

ui_precision_derating
ui_precision_power
ui_precision_sensitivities
ui_precision_timing
unassign_bumps
unassign_tsv
undo_uniquify_partition
unflatten_ilm
ungroup
uniquify
uniquify_filename
uniquify_partition
unlock_oa_design
unplace_obj
update_analysis_view
update_bump_connect_target_constraints
update_bump_name
update_bus_guide
update_clock_tree_source_latency
update_clock_tree_spec_annotations
update_constraint_mode
update_delay_corner
update_floorplan
update_floorplan_obj
update_group
update_inbound_cell_site
update_inst
update_io_latency
update_lef_macro
update_library_set
update_metric
update_metric_definition
update_module_model
update_names
update_net_group
update_oa_lib
update_obj_floorplan_cut_box
update_opcond
update_origin
update_partition
update_partition_cell_obs
update_partition_cut

update_partition_to_stairway
update_pin_group
update_pipeline_net_group
update_power_domain
update_power_switches
update_power_vias
update_rc_corner
update_route_rule
update_sdp
update_skew_group
update_timing_budget
update_timing_condition
update_vertical_cell_edge_constraint
usf_version

V

view_last
view_log
view_next
view_package_results

W

write_annotated_transition
write_black_box_lef
write_bump_locations
write_cap_table
write_category_summary
write_codesign_die_abstract
write_color_preference
write_combinational_loop_break_constraints
write_congested_area_report
write_cts_config
write_current_region
write_db
write_db_auto_save_user_globals
write_db_binary_timing_constraints
write_db_cmd_file_limit
write_db_copy_timing_constraints_always
write_db_create_read_file
write_db_include_metal_fill_rules
write_db_save_unused_lef_block_names
write_def

write_def_by_section
write_def_compress_vias
write_def_hierarchy_delimiter
write_def_include_lef_ndr
write_def_include_lef_vias
write_def_lef_out_version
write_def_polygon_die_area
write_def_stream_check_uncolored
write_design
write_design_stack_config
write_die_abstract
write_die_model
write_do_lec
write_drc_markers
write_exclude_nets
write_extraction_spec
write_flip_chip_property
write_floorplan
write_floorplan_script
write_flow
write_flow_template
write_generalized_feedthru_paths
write_global_slack_report
write_hier_view
write_hinst_colors
write_ilm
write_ilm_eco_db
write_integration_route_constraints
write_io_file
write_ldb
write_lec_dft_constraints
write_lec_directory_naming_style
write_lec_files
write_lef_abstract
write_lef_abstract_customer_header
write_lef_library
write_litho_hotspots
write_metric
write_metric_config
write_micro_bump_mapping_file
write_ml_design
write_ml_library

write_ml_timing_graph
write_ml_timing_path
write_mmmc
write_module_model
write_module_model_context
write_multi_bit_flop_mapping_file
write_name_mapping
write_net_groups_topo
write_netlist
write_netlist_full_pin_out
write_netlist_port_association_style
write_nets_in_congested_area
write_oa
write_oa_abstract
write_oa_black_boxes
write_oa_techfile
write_parasitics
write_partition_pins
write_partitions
write_path_categories
write_path_descriptions
write_path_list_summary
write_pg_library
write_power_constraints
write_power_intent
write_power_switches
write_power_up_report
write_preserves
write_rcdb
write_repeater_rules
write_route_rule
write_sdc
write_sdf
write_sdp_file
write_set_load
write_spef_between_tsv_dies
write_stream
write_stream_allow_path_type4
write_stream_cell_instance_color
write_stream_cell_master_color
write_stream_cell_name_prefix
write_stream_cell_name_suffix

write_stream_check_map_file
write_stream_compatible
write_stream_define_via_name
write_stream_ignore_fixed_mask
write_stream_label_all_pin_shape
write_stream_merge_append
write_stream_oasis_cell_offset
write_stream_oasis_compression
write_stream_oasis_layer_name
write_stream_pin_text_orientation
write_stream_remove_nets
write_stream_snap_to_mfg
write_stream_stream_convert_rect_to_path
write_stream_stream_version
write_stream_text_size
write_stream_uniquify_cell_names_prefix
write_stream_via_names
write_stream_virtual_connection
write_target_based_opt_file
write_tcf
write_tech_pg_lib
write_text_timing_report
write_timing_budgets
write_timing_model
write_timing_windows
write_to_gif
write_usf
write_via_defs
write_workspace
write_xpgv

Z

zoomTo